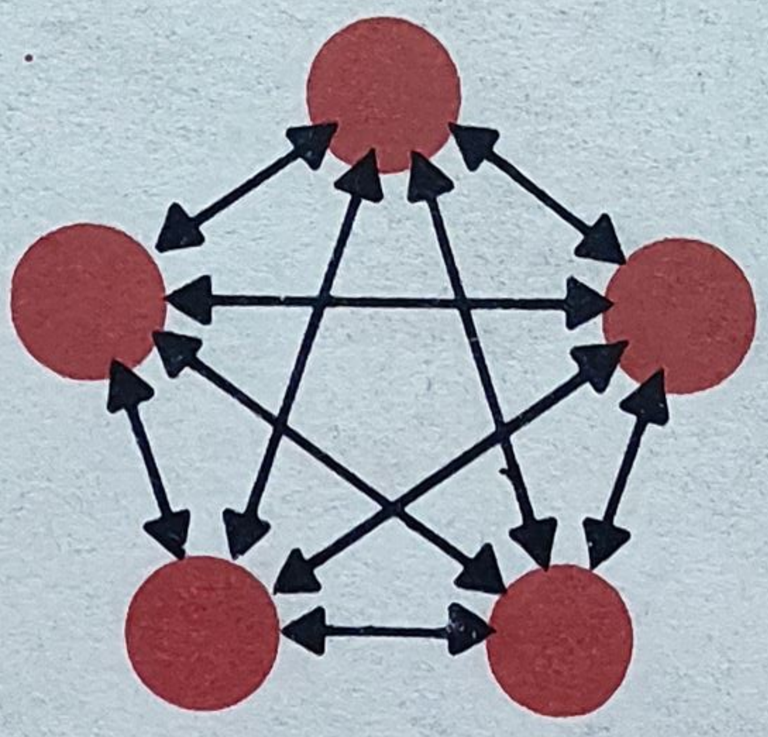
LUCIA COCULESCU
VALENTIN CRISTEA
IOAN FINTA
VICTOR PATRICIU
FLORIN PILAT



PROIECTAREA SISTEMELOR TELEINFORMATICE

EDITURA MILITARĂ

LUCIA COCULESCU VALENTIN CRISTEA IOAN FINTA
VICTOR PATRICIU FLORIN PILAT

PROIECTAREA SISTEMELOR TELEINFORMATICE

EDITURA MILITARĂ, BUCUREȘTI, 1988

PREFAȚĂ

Analiza dezvoltării economice și sociale în multe din țările lumii atestă ponderea deosebită a ramurilor industriale noi, de înaltă tehnicitate, și în mod deosebit a electronicii, microelectronicii, automaticii și tehnicii de calcul care dețin un rol determinant în afirmarea revoluției științifice și tehnice contemporane.

În România, progresul științei și tehnicii a devenit parte componentă a eforturilor pentru edificarea societății socialiste multilateral dezvoltate, un factor deosebit de important în dezvoltarea bazei tehnico-materiale a economiei naționale. Referindu-se la direcțiile de acțiune pentru realizarea sarcinilor revoluției socialiste în noua etapă de dezvoltare a României, tovarășul Nicolae Ceaușescu, secretar general al Partidului Comunist Român, aprecia că „Trebuie să acționăm cu toată hotărîrea în direcția intensificării activității de cercetare, pentru legarea și mai strînsă a cercetării cu producția și învățămîntul, soluționarea într-un termen scurt a problemelor multiple și complexe ale progresului tehnic, ale introducerii rapide în producție, în toate sectoarele, a noilor realizări ale științei și tehnicii“ [1]. *Această cerință sintetizează direcțiile de efort îndreptat spre creșterea continuă a rolului științei și tehnologiei ca forță nemijlocită de producție, ca factor dinamizator al ridicării patriei noastre pe noi trepte de progres și civilizație.*

Avînd în vedere implicațiile sociale ale utilizării calculatoarelor electronice, în țara noastră politica de dotare cu tehnică de calcul este urmărită cu deosebită grijă, pentru a asigura o corelare optimă cu dezvoltarea în ansamblu și utilizarea acestei tehnici la parametri superiori. De la înalta tribună a Congresului științei și învățămîntului tovarășul Nicolae Ceaușescu, secretar general al Partidului Comunist Român, arăta că „Este necesar să se acorde o mai mare atenție

---

[1] Nicolae Ceaușescu, *Raport la Conferința Națională a Partidului Comunist Român*, 14—16 decembrie 1987, Editura Politică, București, 1987, p. 25.

problemelor privind organizarea științifică a producției și a muncii, automatizării, electronizării și robotizării producției, creșterii mai puternice a productivității muncii sociale" [2]. *Realizările industriei de calculatoare din ultimii ani au permis dotarea accelerată a economiei în cadrul programului național și înscrierea României în rîndul țărilor exportatoare de tehnică de calcul. Gama de produse ce se fabrică în prezent s-a înnoit și diversificat în raport cu ceilalți ani, răspunzînd deziderateloi actuale ale tehnicii: modernizarea și realizarea unor tipuri noi de produse, mărirea vitezei de calcul, creșterea productivității.*

Ideea de a lega între ele, la distanță, mai multe calculatoare care să lucreze împreună în cadrul unor sisteme teleinformatice este relativ recentă, putînd fi localizată în timp cu aproximație la mijlocul anilor '60. Punerea în practică a acestei idei a cunoscut însă o evoluție spectaculoasă datorită progreselor rapide înregistrate, pe de o parte, de tehnologia comunicațiilor și, pe de altă parte, de cea a calculatoarelor. În primul caz prin utilizarea celor mai diverse medii de transfer (rețele telefonice, cabluri coaxiale, fibre optice, unde radio etc.) s-a realizat transferul datelor cu viteze din ce în ce mai mari și, ceea ce este la fel de important, cu o fiabilitate din ce în ce mai bună. În cel de al doilea, prin microminiaturizare pe scară largă, s-au creat condițiile pentru răspîndirea geografică mai mare a echipamentelor „inteligente" aparținînd aceluiași sistem și, ca urmare, pentru distribuirea mai puternică a funcțiilor de prelucrare a datelor.

Utilizarea rețelelor de calculatoare (care constituie suportul sistemelor teleinformatice) a permis rezolvarea unor probleme greu sau deloc abordabile cu alte mijloace, pentru care accesul la distanță și distribuirea puterii de calcul reprezintă condiții esențiale: conducerea producției, dirijarea traficului, supravegherea distribuției energiei și a combustibililor, rezervarea locurilor etc. Totodată, perfecționarea rețelelor de calculatoare a stimulat utilizatorii, generînd noi aplicații cum sînt cele telematice, orientate spre probleme curente ale publicului larg. O astfel de perfecționare nu ar fi fost posibilă fără o abordare sistematică și bine fundamentată teoretic a conceperii și realizării sistemelor teleinformatice, putîndu-se astăzi vorbi de o tehnologie specifică acestor sisteme. Unele din aspectele sale cele mai semnificative sînt cuprinse în lucrarea de față.

Autorii pornesc de la ideea, larg verificată în practica proiectării, că abordarea oricărui sistem complex, așa cum este un sistem teleinformatic, este mult facilitată de divizarea sa în subsisteme mai simple și de studiul separat al subsistemelor și al relațiilor dintre ele. Pentru ca o astfel de împărțire să fie utilă, ea trebuie făcută după un criteriu convenabil scopului urmărit. În lucrare se prezintă arhitectura generală a sistemelor teleinformatice prin prisma unui criteriu larg acceptat astăzi de specialiști, acela al funcțiilor realizate de subsistemele componente. Se obțin astfel imagini ale sistemelor teleinformatice alcă-

---

[2] Nicolae Ceaușescu, *România pe drumul construirii societății socialiste multilateral dezvoltate*, vol. 29, Editura Politică, București, 1987, p. 190.

tuite din ierarhii de niveluri funcţionale. Aceasta simplifică identificarea principalelor probleme de proiectare şi permite sistematizarea expunerii metodelor de tratare şi a rezultatelor obţinute prin aplicarea lor.

Orice activitate de proiectare presupune realizarea unui model al obiectului proiectat. Deoarece sistemele teleinformatice sînt în esenţă sisteme cu activităţi paralele asincrone, problemele de proiectare pot fi abordate pe baza a două categorii de modele: cele de calcul paralel şi, respectiv, cu cozi de aşteptare.

În lucrare, autorii prezintă sintetic modelele de calcul paralel, oprindu-se la acelea a căror utilizare în proiectarea protocoalelor pentru sisteme teleinformatice s-a dovedit fructuoasă. Sînt prezentate soluţii interesante de coordonare a activităţilor, ca acelea de cooperare între procesele unui sistem inteligent (distribuit) de rezolvare a problemelor. După o topică similară se prezintă aspectele referitoare la a doua categorie de modele, exemplificîndu-se utilizarea lor în proiectarea topologică a sistemelor teleinformatice.

În demersul lor analitic, autorii pun în evidenţă etapele de realizare a unui sistem teleinformatic: analiză preliminară şi specificare funcţională, proiectare de ansamblu, proiectare de detaliu, elaborare de programe, integrare şi testare, punere în funcţiune/experimentare şi acceptarea sistemului teleinformatic.

Ei prezintă conţinutul acestor etape, problemele, metodele şi instrumentele de tratare, precum şi rezultatele remarcabile obţinute prin aplicarea lor în cazul unor sisteme teleinformatice concrete.

În multe aplicaţii, securitatea informaţiei reprezintă o cerinţă importantă pe care proiectantul unui sistem teleinformatic trebuie să o aibă în vedere. Numărul mare de utilizatori care lucrează simultan cu sistemul, precum şi repartiţia geografică a acestuia amplifică cerinţele de control al accesului la resurse şi de protecţie a informaţiei. Metodele şi tehnicile utilizate în autorizarea accesului în sistem, autentificarea utilizatorilor, realizarea semnăturii digitale, protecţia fişierelor şi a bazelor de date sînt analizate concis şi sistematic în lucrare.

O problemă majoră în realizarea unui sistem teleinformatic o constituie atingerea unor indici de performanţă superiori, idee urmărită pe durata întregului său ciclu de viaţă. Lucrarea prezintă principalele metode şi tehnici de evaluare şi optimizare prin modelare, simulare şi măsurători realizate în etapele de proiectare, implementare şi întreţinere. O atenţie deosebită este acordată optimizării topologice a sistemelor teleinformatice şi reconfigurării acestora în cazul defectării unei părţi sau la schimbarea traficului datelor. Sînt făcute aprecieri şi asupra eficienţei economice de realizare a sistemelor teleinformatice.

Axată pe o problematică actuală, interesantă pentru un număr mare de specialiști și utilizatori ai tehnicii de calcul, lucrarea poate contribui la creșterea calității activității proiectanților sistemelor teleinformatice și a celor ce folosesc sau doresc să folosească astfel de sisteme.

Bazîndu-se pe un bogat material bibliografic, pe experiența acumulată în decursul unei activități didactice îndelungate și pe rezultatele obținute în cercetările proprii, autorii reușesc să prezinte coerent, modern și cu multe elemente de originalitate domeniul proiectării sistemelor teleinformatice care va deveni, probabil, unul din subiectele centrale ale societății informatizate a anilor 2000.

Dr. ing. VASILE BALTAC

CAPITOLUL 1

ARHITECTURA SISTEMELOR TELEINFORMATICE

Sistemele teleinformatice se bucură azi de atenția unor largi cercuri de specialiști și utilizatori. Răspîndirea lor din ce în ce mai mare, inclusiv în țara noastră, se datorează în principal avansului tehnologic înregistrat în două domenii de vîrf ale tehnicii: calculatoarele și telecomunicațiile. Calculatoarele permit prelucrarea mai rapidă a unor cantități sporite de informație; telecomunicațiile oferă suportul transferului datelor la distanțe mari, făcînd ca informația să fie ușor accesibilă în cele mai diferite locuri în care oamenii își desfășoară activitatea. Totodată, dezvoltarea sistemelor teleinformatice este garantată de aplicații. Nu numai că sistemele teleinformatice amplifică performanțele sistemelor informatice, prin economicitate, o mai mare comoditate de exploatare și o disponibilitate crescută, dar ele satisfac cerințe noi, din ce în ce mai sofistificate ale utilizatorilor. Avînd la bază sisteme de calcul distribuite ca funcții, prelucrări, resurse și control, sistemele teleinformatice sînt de neînlocuit în aplicații care ele însele sînt — prin natura lor — distribuite. Este greu să ne imaginăm astăzi desfășurarea unor activități, cum ar fi rezervarea de locuri pentru companiile de transport aerian, conducerea operativă a traficului pe calea ferată sau în marile stații de triaj, conducerea proceselor de producție în cadrul unor combinate ori platforme industriale, controlul distribuției energiei și combustibilului prin rețeaua energetică, respectiv prin rețeaua de livrare a gazului metan, precum și a multor altor activități, în afara unor sisteme teleinformatice.

1.1. CONCEPTE DE BAZĂ

Sistemele teleinformatice au configurații foarte diverse, de la cele mai simple, cuprinzînd cîteva terminale legate la distanță față de un sistem de calcul central, la cele mai complexe, avînd sute și chiar mii de terminale și calculatoare interconectate prin intermediul unui subsistem de comunicație la distanță. Deși răspîndirea geografică a componentelor este caracteristică sistemelor teleinformatice, ea nu reprezintă singura proprietate pe care specialiștii trebuie să o aibă în vedere la proiectarea și realizarea unor astfel de sisteme. În continuare menționăm unele caracteristici ale sistemelor teleinformatice pe care le considerăm ca fiind cele mai importante.

Sistemul teleinformatic include un număr arbitrar de procese, care pot fi procese de sistem sau procese ale utilizatorilor. Fiecare proces este „găzduit" de un sistem de calcul numit gazdă. Sistemul gazdă conține fie mai multe procesoare fie unul singur, care este — eventual — multiplexat între mai multe procesoare virtuale. De asemenea, este posibil ca mai multe sisteme gazdă să împartă în comun același procesor. Sistemul gazdă are una sau mai multe interfețe de legătură cu rețeaua de comunicație.

Comunicarea între procese se face prin mesaje, avînd ca suport un subsistem de comunicație distribuit. Acesta asigură ca două procese să comunice între ele în același mod, indiferent dacă sînt localizate într-un singur sistem gazdă sau în sisteme gazdă situate la distanță unul față de altul.

Întîrzierea de transmitere a mesajelor este variabilă și diferită de zero. Această caracteristică, fundamentală pentru sistemele teleinformatice, are implicații directe asupra modului de concepere și de realizare a cooperării diferitelor procese componente.Ea diferențiază sistemele teleinformatice de cele cu control centralizat, în care fiecare proces este capabil să obțină o imagine consistentă și completă a stării sistemului. Datorită caracterului distribuit al activităților (determinat de distribuția geografică a sistemelor gazdă) fiecare proces dispune de o imagine parțială a sistemului. Mai mult, pentru a corela între ele imaginile deținute de diferite procese sînt necesare mecanisme de control specifice, cu acțiune la nivelul întregului sistem teleinformatic.

Arhitectura sistemelor teleinformatice este modulară și structurată ierarhic. Această caracteristică rezultă din faptul că sînt concepute, proiectate și realizate sistematic. Proiectarea oricărui sistem teleinformatic este mult simplificată prin divizarea lui în module și prin studiul separat al modulelor și al relațiilor dintre ele. Pentru ca să fie utilă o astfel de împărțire trebuie să aibă la bază un criteriu convenabil scopului urmărit. Evident, utilizarea unor criterii diferite poate conduce la împărțiri diverse, obținîndu-se „imagini" distincte ale aceluiași sistem. În continuare arhitectura generală a unui sistem teleinformatic va fi privită prin prisma unui criteriu larg acceptat astăzi de specialiști, și anume acela al funcțiilor realizate de subsistemele componente. Se obține astfel o imagine a sistemului teleinformatic alcătuită dintr-o ierarhie de niveluri funcționale. Aceasta simplifică identificarea principalelor probleme de proiectare specifice sistemelor teleinformatice și permite sistematizarea expunerii metodelor de tratare a rezultatelor obținute prin aplicarea lor. Structurarea ierarhică constituie premisa necesară extinderii sistemelor teleinformatice, prin adăugarea unor funcții noi sau prin modificări cerute de îmbunătățirea performanțelor. Chiar instalarea unui sistem modular este mai simplă: se poate porni de la o configurație inițială redusă, avînd un cost scăzut, restul modulelor fiind adăugate ulterior, în mod treptat, cu un efort mai mic de punere la punct.

Datorită caracteristicilor menționate, sistemele teleinformatice permit atingerea unor obiective importante pentru realizatorii și utilizatorii lor, cum ar fi creșterea performanțelor și a disponibilității.

1.2. NIVELURI ARHITECTURALE

Așa cum s-a arătat, un sistem teleinformatic are ca suport o colecție de calculatoare interconectate prin intermediul unui subsistem de comunicație, iar funcționarea sa presupune cooperarea între toate componentele în vederea satisfacerii solicitărilor utilizatorilor. Deoarece au un grad de complexitate ridicat, funcțiile sistemelor teleinformatice sînt realizate prin compunerea unor funcții mai simple. Un sistem apare astfel ca o ierarhie de niveluri funcționale. Fiecare nivel N furnizează nivelurilor superioare un set bine definit de servicii, pentru realizarea cărora se bazează pe serviciile nivelurilor inferioare. Structura internă și mecanismele caracteristice nivelului N nu sînt vizibile celorlalte niveluri, fiind importante doar serviciile oferite, precum și regulile de acces la ele (cu alte cuvinte interfața nivelului N cu restul nivelurilor). Aceasta reprezintă singura modalitate de interacționare a nivelurilor (fig. 1.1.).

Funcțiile unui nivel sînt realizate prin acțiunile corelate ale modulelor componente, ce cooperează între ele în conformitate cu un anumit set de reguli numit protocol. Structura unui modul nu este vizibilă în exterior, fiind important doar felul în care acesta interacționează cu celelalte module ale nivelului.

În domeniul sistemelor de calcul ideea de stratificare nu a apărut odată cu sistemele teleinformatice. Ea a fost la început utilizată cu succes în alcătuirea modelelor sistemelor de operare și apoi a celor ale sistemelor informatice mai complexe. Era însă firesc ca un asemenea principiu să fie folosit și la proiectarea și implementarea sistemelor teleinformatice. În primul rînd pentru că înțelegerea unui astfel de sistem este facilitată de împărțirea sa în componente mai simple. Prin aceasta, soluțiile adoptate la proiectare pot fi ușor evaluate, selectîndu-se variantele cele mai bune. Apoi, procesul de implementare și de punere la punct a sistemului este înlesnit prin adoptarea unor metodologii specifice arhitecturilor stratificate. Implementarea corespunzătoare unui anumit nivel sau modul poate diferi de la un sistem de calcul la altul (fiind concepută astfel încît să folosească cît mai bine particularitățile acestora), cu condiția ca serviciile oferite la interfață să fie identice. În fine,

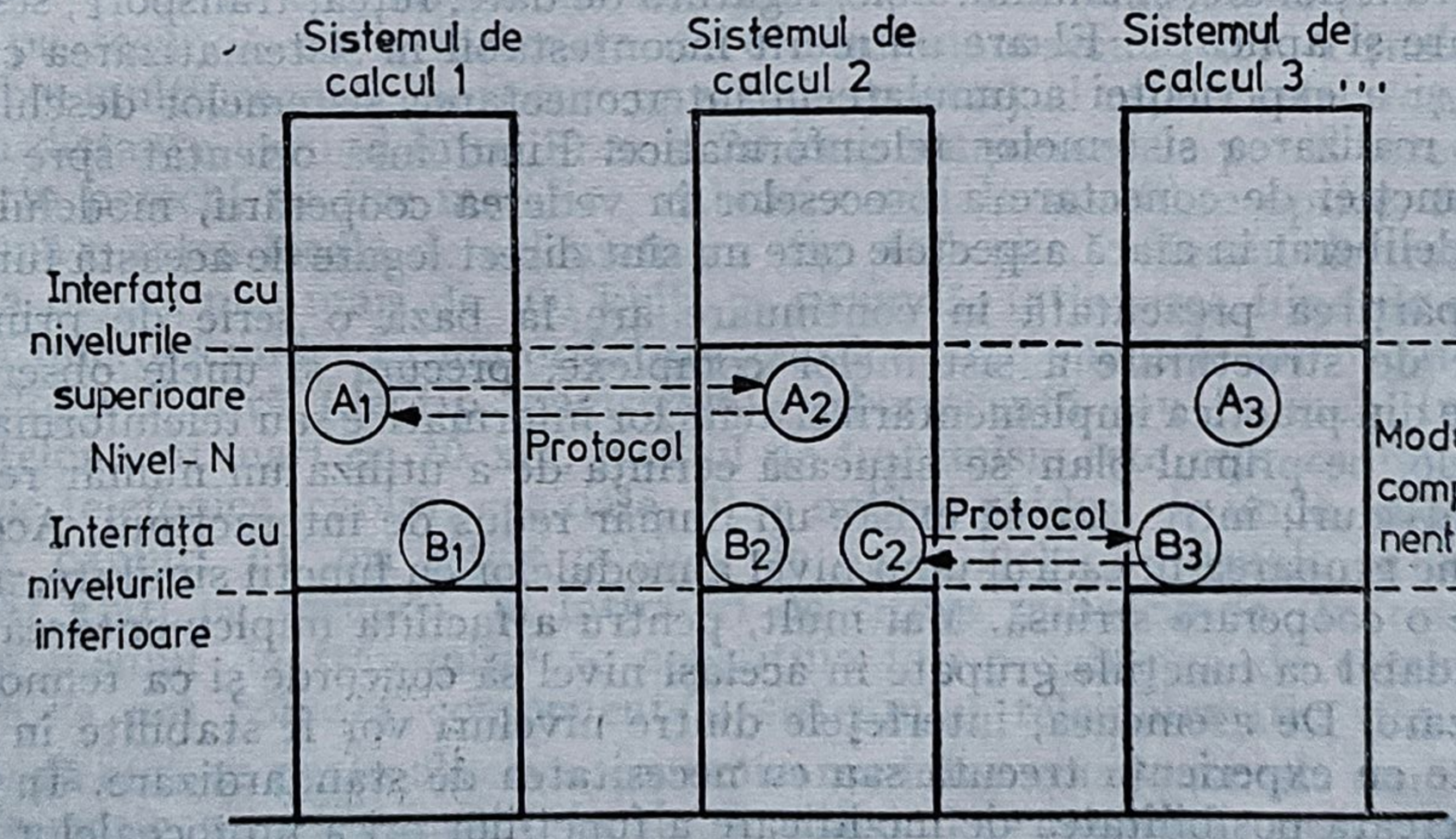


Fig. 1.1. Stratificarea funcțiilor sistemelor teleinformatice

sistemul este ușor de întreținut, deoarece există posibilitatea ca oricare nivel ori subnivel să fie înlocuit cu altul mai performant, dar care oferă nivelurilor superioare același set de servicii.

În procesul descompunerii funcționale a unui sistem teleinformatic, nivelul, subnivelul și modulul sînt privite în mod uniform, prin prisma comportării observate la interfața lui cu exteriorul. Unii autori [L-1.] utilizează chiar o terminologie comună pentru aceste elemente, considerînd că serviciile furnizate de nivelurile și modulele componente ale unui sistem teleinformatic pot fi precizate prin definirea unor obiecte abstracte accesibile doar prin interfețe bine stabilite. Orice tip de obiect este specificat (asemenea tipurilor abstracte de date) prin reprezentarea sa — constituită de structurile de date vizibile la interfața obiectului — și prin operațiile permise asupra reprezentării. Unui obiect îi pot corespunde mai multe implementări diferite, care coexistă în același sistem teleinformatic, deoarece amănuntele de realizare nu sînt semnificative în exteriorul obiectelor.

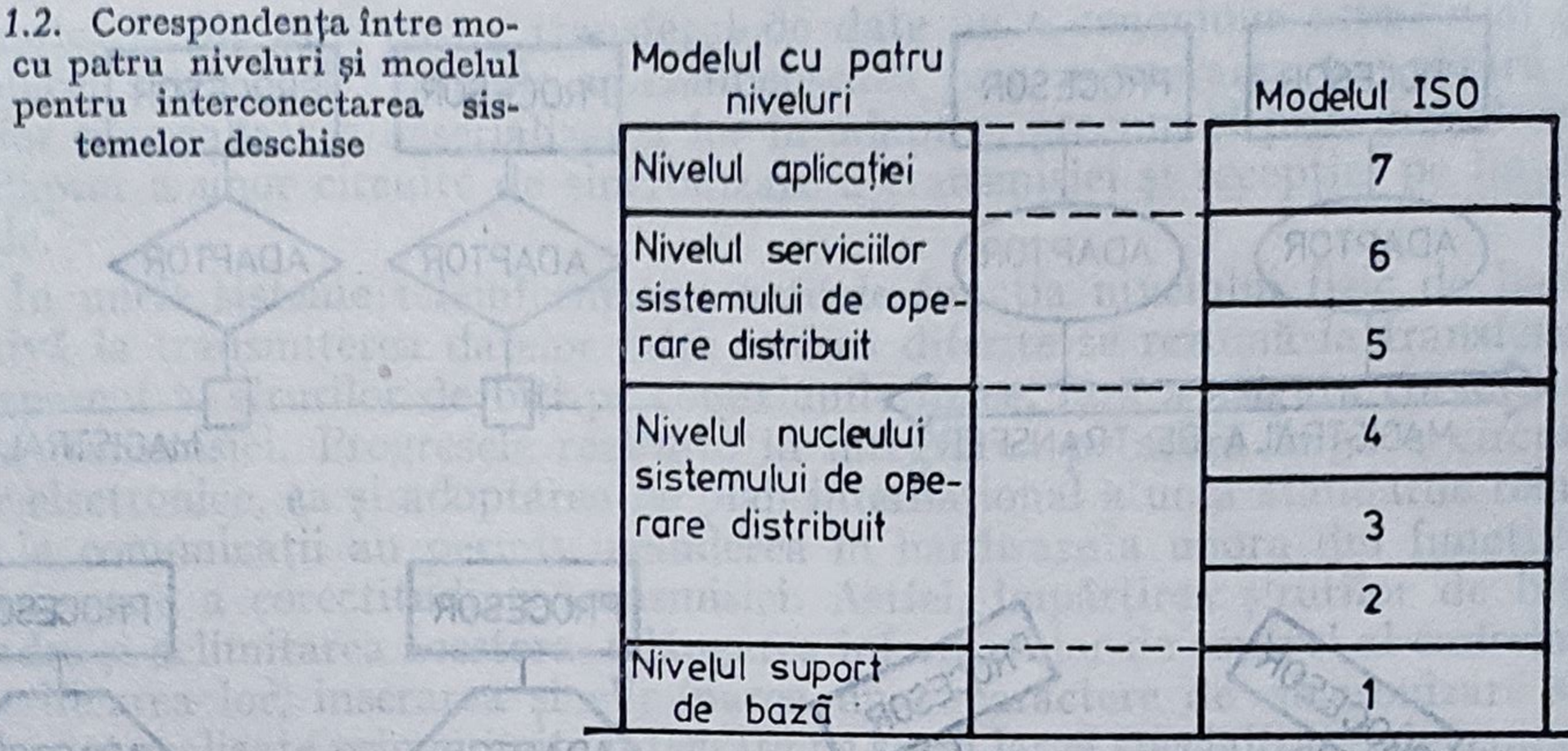
Sistemul teleinformatic poate fi văzut ca o ierarhie de obiecte, la fiecare nivel obiectele fiind concepute pe baza celor de pe nivelurile inferioare. De asemenea, sistemul are posibilitatea să fie extins dacă sînt definite obiecte noi pe baza celor deja existente. Se disting două categorii de obiecte: obiecte active, care corespund proceselor de calcul, și obiecte pasive (cum ar fi cele corespunzătoare echipamentelor de intrare/ieșire, canalelor de comunicație, mesajelor, fișierelor etc.) asupra cărora acționează procesele pentru a le schimba starea.

Împărțirea conceptuală a funcțiilor în niveluri și subniveluri permite obținerea mai multor modele arhitecturale pentru un același sistem. Ele pot să-l prezinte cu grade de detaliere diferite, grupează funcțiile în moduri deosebite, după criteriul avut în vedere la modelare, evidențiază interacțiuni mai complexe sau mai simple între modelele și nivelurile componente.

Unul dintre modelele cele mai cunoscute de stratificare a funcțiilor rețelelor de calculatoare este cel de referință pentru interconectarea sistemelor deschise elaborat de ISO și descris în detaliu în lucrarea [P-1.]. Acest model împarte funcțiile relative la interconectarea sistemelor de calcul în șapte niveluri funcționale, și anume: fizic, legătură de date, rețea, transport, sesiune, prezentare și aplicație. El are un merit incontestabil în sistematizarea cunoștințelor și a experienței acumulate în interconectarea sistemelor deschise și, deci, în realizarea sistemelor teleinformatice. Fiind însă orientat spre detalierea funcției de conectare a proceselor în vederea cooperării, modelul lasă în mod deliberat în afară aspectele care nu sînt direct legate de această funcție.

Împărțirea prezentată în continuare are la bază o serie de principii generale de structurare a sistemelor complexe, precum și unele observații rezultate din practica implementării sistemelor informatice sau teleinformatice. Dintre ele, pe primul plan se situează cerința de a utiliza un număr relativ mic de niveluri, între care să existe un număr redus de interacțiuni. Aceasta presupune gruparea în cadrul unui nivel a modulelor cu funcții similare, avînd între ele o cooperare strînsă. Mai mult, pentru a facilita implementarea este recomandabil ca funcțiile grupate în același nivel să concorde și ca tehnologie de realizare. De asemenea, interfețele dintre niveluri vor fi stabilite în conformitate cu experiența trecută sau cu necesitatea de standardizare. În fine, trebuie creată posibilitatea de modificare a funcțiilor ori a protocoalelor unui nivel fără a afecta nivelurile adiacente. Aplicarea acestor principii a condus

Fig. 1.2. Corespondența între modelul cu patru niveluri și modelul ISO pentru interconectarea sistemelor deschise



la obținerea unui model cu patru niveluri funcționale, și anume [L-1.]: nivelul suport de bază, nivelul nucleului sistemului de operare distribuit, nivelul serviciilor sistemului de operare distribuit și nivelul aplicației. Modelul include funcțiile de interconectare a proceselor, evidențiate de modelul ISO pentru interconectarea sistemelor deschise, putîndu-se pune chiar în corespondență cele două modele, așa cum se arată în figura 1.2. Dăm în continuare o descriere succintă a funcțiilor fiecăruia din cele patru niveluri enumerate.

Nivelul suport de bază realizează operațiile (funcțiile) primitive de memorare, de prelucrare și de transfer al informației. El include procesoare (unități centrale de prelucrare), memorii, unități de intrare/ieșire, terminale, echipamente de culegere de date și de control al proceselor industriale etc., interconectate printr-un mediu de comunicare. Răspîndirea geografică a componentelor, numite în continuare și noduri, determină mijloacele de interconectare și, totodată, nivelurile atinse de debitul datelor și de întîrzierea de transfer. Pentru distanțe de cîțiva metri, componentele pot fi conectate printr-o magistrală globală de transfer, formînd împreună un sistem multiprocesor (fig. 1.3.a). Magistrala conține linii diferite pentru transferul datelor, al adreselor și al semnalelor de control și permite transmiterea a cîtorva Mbiți pe secundă cu o întîrziere de ordinul nanosecundelor. Uzual, acest ansamblu multiprocesor constituie un sistem (unitar) al rețelei teleinformatice. Pentru distanțe variind între 10 m și 20 km, componentele pot fi interconectate prin perechi de fire metalice, cabluri coaxiale sau fibre optice, formînd împreună o rețea locală de calculatoare (fig. 1.3b și c.). Mediul de transmisie suportă debite mai mari de 100 kbiți pe secundă, întîrzierea fiind de ordinul cîtorva microsecunde. În majoritatea cazurilor componentele sînt legate după o topologie liniară (de tip magistrală), arborescentă sau în inel. În cazul distanțelor mai mari de 20 km, mediul de transmisie este reprezentat de fire metalice telefonice, cabluri coaxiale, fibre optice, ghiduri de undă, unde radio, sateliți, cabluri TV ori rețele publice de date. Debitele uzuale se situează sub 100 kbiți pe secundă, la întîrzieri de cîteva milisecunde. În realizările recente mediul de transmisie se constituie într-o subrețea de comunicație la care se pot conecta componentele sistemului teleinformatic (fig. 1.3d).

După cum se poate observa, comunicarea între componentele unui sistem teleinformatic se realizează prin două tipuri de conexiuni: punct-la-punct

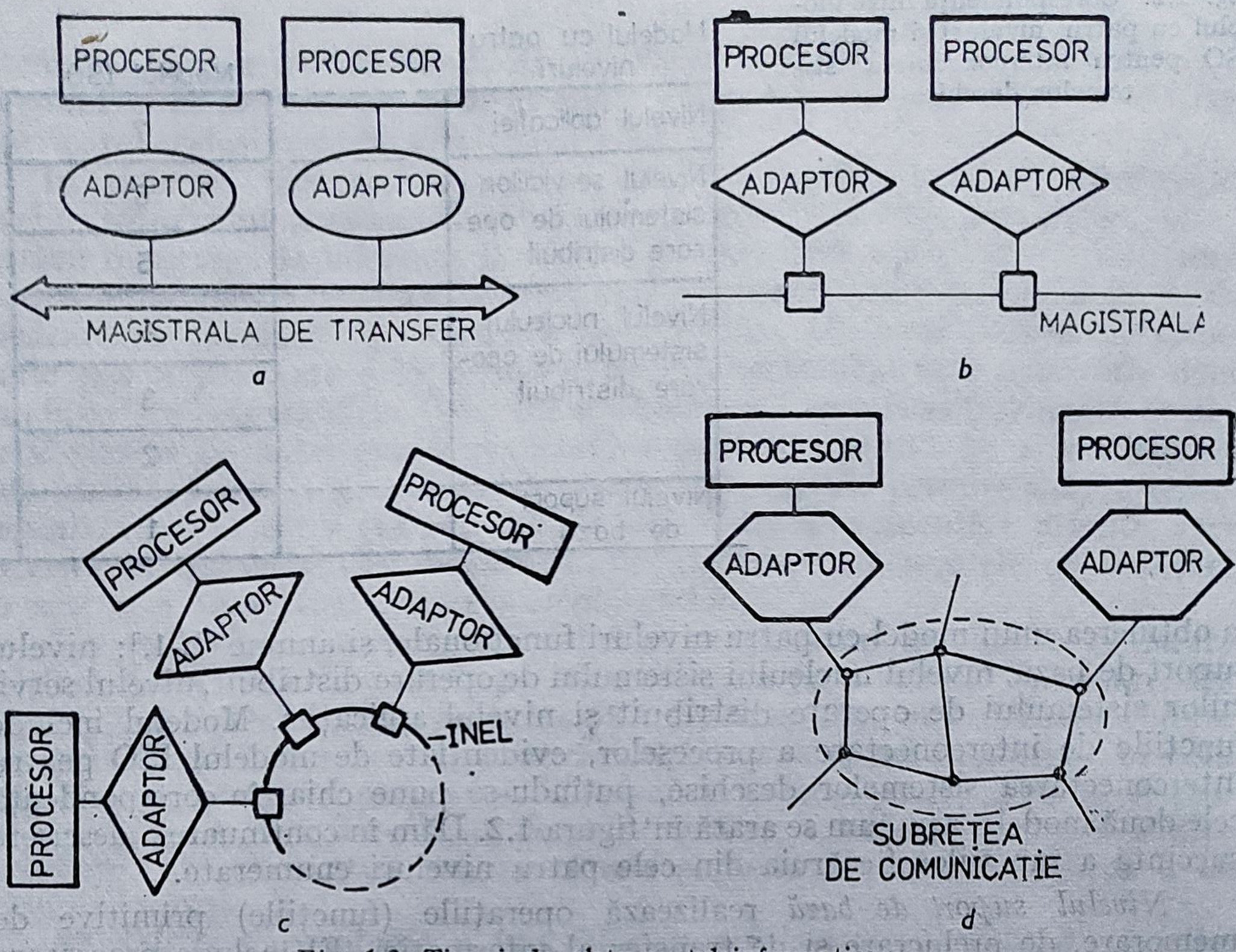


Fig. 1.3. Tipuri de sisteme teleinformatice:
a — sistem multiprocesor; *b* — rețea locală cu topologie liniară; *c* — rețea locală în inel; *d* — rețea larg răspîndită geografic

(fig. 1.3d) sau multipunct (fig. 1.3b și c). În primul caz conexiunea leagă între ele două noduri ale sistemului teleinformatic între care informația este transmisă direct. Două noduri ce nu sînt legate printr-o conexiune directă își pot comunica informații prin mai multe conexiuni și noduri intermediare. O conexiune multipunct leagă un nod cu mai multe alte noduri diferite. Ea permite transmiterea informației între două noduri conectate oarecare, dar și transmiterea difuzată a informației către mai multe noduri simultan. Utilizarea unei conexiuni multipunct ridică o problemă suplimentară, pe care nu o întîlnim la conexiunile punct-la-punct, și anume aceea a determinării nodului care are dreptul să transmită la un moment dat (alocarea conexiunii). Oricum, într-un sistem teleinformatic pot fi utilizate ambele tipuri de conexiuni, în combinații ce conduc la topologii foarte diverse.

Cuplarea unui nod la mediul de comunicație se face prin intermediul unui adaptor. Rolul său principal este de a gestiona conectarea nodului la mediu, asigurînd astfel integrarea sa armonioasă în rețea pe durata funcționării. De exemplu, în cazul cuplării la o rețea telefonică prin intermediul unui modem adaptorul îndeplinește toate funcțiile de control direct al modemului. Totodată, adaptorul realizează compensarea diferențelor între modul în care sistemele de calcul schimbă informații cu exteriorul și modul de transfer prin mediul de comunicație. Astfel, un sistem de calcul transmite și recepționează informații prin magistrale de intrare/ieșire, utilizînd canale de acces la memorie sau sistemul de întreruperi. Ca urmare, transferul de date este asincron și se face în paralel pentru toți biții unui octet sau ai unui cuvînt de

memorie. Pe de altă parte, transferul de date pe o conexiune este serial și de obicei sincron. Diferențele se compensează prin memorarea temporară a datelor și serializarea/deserializarea lor în adaptor, precum și prin includerea în adaptor a unor circuite de sincronizare a transmisiei și recepției pe liniile seriale.

În unele sisteme teleinformatice actuale funcția nivelului fizic de bază relativă la transmiterea datelor între noduri diferite se rezumă la transferul transparent al șirurilor de biți pe conexiunile fizice, fără a asigura corectitudinea transmisiei. Progresele realizate în integrarea pe scară largă a circuitelor electronice, ca și adoptarea pe plan internațional a unor standarde relative la comunicații au permis includerea în hardware a unora din funcțiile de asigurare a corectitudinii transmisiei. Astfel, împărțirea șirurilor de biți în cadre și delimitarea acestora, adăugarea informațiilor de control al cadrelor și verificarea lor, inserarea și eliminarea unor caractere de sincronizare și altele sînt realizate prin circuite integrate pe scară largă specializate, devenind în felul acesta funcții ale nivelului fizic de bază.

Răspîndirea tot mai mare a sistemelor teleinformatice impune aducerea unor îmbunătățiri prin care componentele acestora să fie tot mai bine adaptate utilizării în medii distribuite eterogene, bazate pe comunicarea de mesaje. Aceasta ar însemna, în primul rînd, modificarea sistemului de intrare/ieșire, trecîndu-se — de la orientarea sa actuală pe semnale de întrerupere — la transmiterea și recepția de mesaje. De asemenea, includerea în hardware a unor funcții care să asiste modificarea reprezentării datelor ar fi un alt aspect important, vizînd compensarea caracterului eterogen al sistemelor distribuite. Putem adăuga la acestea mecanismele necesare realizării eficiente a protecției, securității și controlului accesului, precum și mecanismele necesare gestiunii informației de stare a sistemului teleinformatic (modificare, testare, distribuire, sincronizare). Mai putem aminti utilizarea mecanismelor de telediagnoză și telemăsură, destinate creșterii fiabilității și disponibilității nodurilor, ca și efectul favorabil pe care includerea unei inteligențe sporite în diferite componente l-ar avea asupra diminuării cerințelor de trafic între noduri și deci asupra performanțelor globale ale sistemelor teleinformatice. Deoarece multe din funcțiile amintite sînt realizate în sistemele teleinformatice actuale de nivelul superior, vom asista probabil la o migrare a funcțiilor către nivelul de bază, pe măsură ce soluțiile de realizare a acestor funcții vor deveni mai bine cunoscute.

Nivelul nucleului sistemului de operare distribuit asigură mecanismele care stau la baza creării obiectelor abstracte (procese, resurse logice), multiplexării componentelor, protecției și securității, furnizînd totodată serviciul de comunicare și sincronizare între procese. Funcțiile realizate de acest nivel pot diferi, fiind dependente atît de nivelul suport de bază, cît și de aplicația pentru care este conceput sistemul teleinformatic. Unii autori (L-1.) consideră că o strategie bună, adaptată structurii eterogene a sistemelor distribuite este de a plasa în nucleu minimul de funcții necesare, incluzînd restul în nivelul superior. Astfel, în cazul unui sistem monoprogramat serviciul de comunicare poate fi asigurat de o componentă a nucleului, în timp ce într-un sistem multiprogramat de mari dimensiuni trebuie să se adauge module care să permită crearea de procese și resurse logice, module pentru controlul accesului la resurse, precum și mecanismele de bază destinate protecției între

procese. Oricum, funcția principală rămîne aceea de asigurare a comunicării între procese, la care ne referim în cele ce urmează.

Comunicarea între două procese ale unui sistem teleinformatic presupune stabilirea unei legături logice, numită și canal logic de comunicație, avînd ca suport unul sau mai multe canale fizice. La funcția de transmitere transparentă a șirurilor de biți pe care o realizează canalul fizic de comunicație legătura logică adaugă o serie de funcții suplimentare. Un proces poate stabili simultan mai multe legături logice, deci are mai multe „porturi" de comunicare cu celelalte procese. Serviciul de comunicație trebuie să asigure posibilitatea ca oricare două procese, indiferent de localizarea lor, să stabilească între ele o legătură logică. Mai mult, modul cum procesele acționează în vederea realizării legăturii și transferului datelor trebuie să fie independent de localizarea proceselor. Singurul element pe care un proces este necesar să-l cunoască pentru a comunica cu un alt proces este identificatorul acestuia, unic la nivelul întregului sistem teleinformatic (bineînțeles, cu respectarea condițiilor impuse de drepturile de acces).

Două procese între care s-a stabilit o legătură logică își pot comunica reciproc informații. Serviciul de comunicație asigură sincronizarea între procese pentru transmiterea și recepția coerentă a mesajelor, încearcă să realizeze transferul datelor fără erori și pune la dispoziția proceselor toate componentele stării curente a schimbului de informații. În scopul evitării degradării funcționării sistemului în cazul producerii unor defecte, fiecare proces păstrează o autonomie și un control complet asupra resurselor și acțiunilor proprii. Astfel, oricare proces este în măsură să anuleze datele pregătite transmisiei și să ignore datele recepționate în așa fel, încît activitatea sa să nu fie afectată de eventuala defectare a procesului cu care a stabilit o legătură logică.

Comunicarea între două procese pe o legătură logică este o funcție complexă ce poate fi privită ca o compunere a unor funcții mai simple. Descrierea detaliată a acestora, realizată în conformitate cu modelul ISO pentru interconectarea sistemelor deschise, se găsește în [P-1.]. Reamintim aici cîteva idei mai importante. O primă funcție este aceea de a asigura că transferul de date pe orice canal fizic de comunicație se desfășoară corect. Pentru aceasta, șirurile de biți transmiși pe canal sînt împărțite în cadre, emițătorul adăugînd fiecărui cadru informații suplimentare de control, pe baza cărora receptorul verifică corectitudinea transmisiei și confirmă cadrele recepționate fără erori. Cadrele recepționate eronat sau pierdute sînt retransmise de emițător. Prin această funcție se creează între două noduri conectate printr-un canal de comunicație o legătură de date, ce permite transferul corect al datelor între cele două noduri, în ambele sensuri.

Datele pot fi transmise corect și între două noduri care nu au o legătură de date directă. Transmisia se face prin mai multe legături și noduri intermediare, fiind necesară — de obicei — alegerea unei rute (combinații de legături) din mai multe posibile. Ca atare, cadrului îi sînt adăugate informații suplimentare de control (adresa nodului-sursă, adresa nodului de destinație etc.), obținîndu-se un pachet care este transferat prin legături succesive de la sursă la destinație. Corectitudinea transferului este asigurată la nivelul fiecărei legături de date. Cu toate acestea, comunicarea sigură între două procese necesită măsuri de verificare și de corecție pentru acele erori ce nu sînt tratate de subnivelurile inferioare. Ele au efect doar în nodurile-sursă

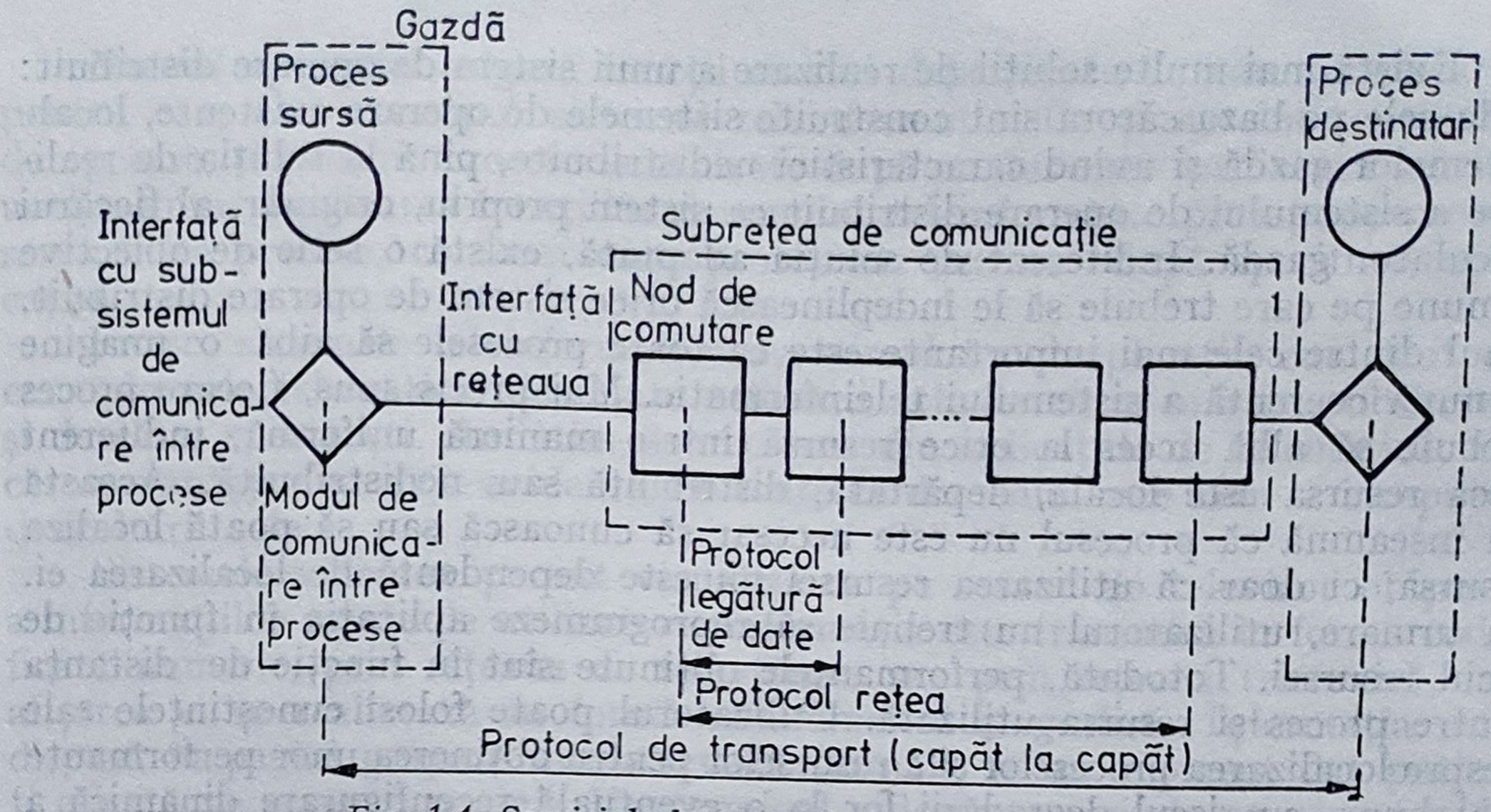


Fig. 1.4. Serviciul de comunicare între procese

și de destinație ale transferului din care cauză sînt denumite funcții capăt-la--capăt, sau funcții de transport. De exemplu, un mesaj mai lung ca să fie transmis trebuie segmentat, fiecare segment fiind încadrat într-un pachet și transmis prin rețea. La nodul destinatar se realizează recompunerea mesajului din pachetele recepționate. În aceeași categorie de funcții se poate considera controlul fluxului, ce asigură o reglementare a ratei cu care datele sînt transportate între procesele-sursă și destinație.

Serviciul de comunicare între procese se bazează pe primitive de transmitere și recepție de mesaje, de control al stării, de anulare a transmisiei, de blocare și de activare a proceselor și funcționează — în linii mari — astfel: analizează informația de adresă a destinatarului, determinînd dacă ea corespunde unui proces local sau îndepărtat. În primul caz sînt utilizate mecanismele de comunicare locală a datelor. În cel de al doilea este folosit serviciul de transport pentru transmiterea la distanță a datelor. Elementele prezentate sînt rezumate schematic în fig. 1.4.

Nivelul serviciilor sistemului de operare distribuit asigură gestiunea resurselor și a proceselor, furnizînd totodată o serie de servicii comune aplicațiilor de diferite categorii. El conține procese dispuse într-o ierarhie de niveluri, după modul în care își furnizează servicii unele altora. Această ierarhie se bazează pe serviciile oferite de nucleul sistemului de operare distribuit și servește aplicația pentru care este conceput sistemul teleinformatic. În general, serviciile oferite aplicației nu diferă de cele pe care un sistem de operare le pune la dispoziția utilizatorilor într-un sistem de calcul nedistribuit. Ele includ următoarele: crearea și distrugerea proceselor; controlul evoluției proceselor, deci garantarea evoluției fiecărui proces pregătit pentru execuție și evitarea blocării definitive a două sau a mai multor procese; alocarea resurselor necesare proceselor pentru execuție, astfel încît performanțele sistemului să fie cît mai bune; asigurarea protecției, securității informației și controlului accesului la resurse, precum și a unor forme diverse de cooperare între procese, utilizînd mecanismele de comunicare furnizate de nucleu. Ceea ce diferă este organizarea internă a sistemului de operare, adică modul în care serviciile enumerate sînt obținute.

Există mai multe soluții de realizare a unui sistem de operare distribuit: de la cele pe baza cărora sînt construite sistemele de operare existente, locale sistemelor gazdă și avînd caracteristici nedistribuite, pînă la soluția de realizare a sistemului de operare distribuit ca sistem propriu, originar, al fiecărui calculator gazdă. Indiferent de soluția adoptată, există o serie de obiective comune pe care trebuie să le îndeplinească orice sistem de operare distribuit. Unul dintre cele mai importante este ca toate procesele să aibă o imagine comună coerentă a sistemului teleinformatic. Mai precis spus, fiecare proces trebuie să aibă acces la orice resursă într-o manieră uniformă, indiferent dacă resursa este locală, depărtată, distribuită sau nedistribuită. Aceasta nu înseamnă că procesul nu este necesar să cunoască sau să poată localiza resursa, ci doar că utilizarea resursei nu este dependentă de localizarea ei. Ca urmare, utilizatorul nu trebuie să reprogrameze aplicația în funcție de locul resursei. Totodată, performanțele obținute sînt în funcție de distanța dintre proces și resursa utilizată. Utilizatorul poate folosi cunoștințele sale despre localizarea proceselor și a resurselor pentru obținerea unor performanțe mai bune, cu riscul degradării lor la o eventuală reconfigurare dinamică a sistemului teleinformatic.

Un al doilea obiectiv este cel al eficienței implementării. Mecanismele adoptate trebuie să facă uz într-o măsură cît mai mare de caracteristicile sistemului suport de bază, dar — în același timp — să ascundă utilizatorului particularitățile diferitelor sisteme sau echipamente, oferindu-i un acces uniform la diversele servicii puse la dispoziția sa. Astfel, accesul unui proces la o resursă se face prin aceleași primitive, indiferent de localizarea resursei; pe de altă parte, însă, dacă resursa este locală atunci accesul trebuie să fie la fel de eficient ca în cazul unui sistem nedistribuit.

În fine, un alt obiectiv este extensibilitatea. Atingerea sa impune includerea în sistemul de operare distribuit a unor mijloace care să permită adăugarea de noi servicii pe baza celor existente, într-o manieră naturală, fără să se efectueze modificări în nivelurile inferioare ale arhitecturii. Aceste mijloace se pot materializa ca un set de procese speciale sau ca rutine de bibliotecă, alcătuind un subnivel important, cu rol de suport al celorlalte servicii ale sistemului de operare, ca și ale aplicației. Realizarea lor impune definirea unor modele a structurilor de date asociate resurselor și proceselor, a interacțiunilor dintre procese și a operațiilor comune unor largi categorii de servicii ale sistemului de operare și ale aplicațiilor.

Modelul tipic pentru o resursă ar putea consta dintr-un ansamblu de structuri de date și un ansamblu de operații definite pe aceste structuri, considerate ca un tot unitar. Orice acțiune de modificare sau testare a datelor caracteristice resursei s-ar face doar prin invocarea operațiilor asociate. Ansamblul datelor resursei conține un antet, cu informații generale pentru orice resursă (de exemplu, data creării, data ultimului acces, drepturile de acces la resursă, nivelul de securitate, date de contabilitate și de performanță etc.), și un corp, cu informații specifice fiecărei resurse în parte.

Similar, modelul unui proces ar putea consta dintr-un ansamblu de structuri de date și un program care se pune în execuție la activarea procesului. Ca și în cazul resursei, datele conțin un antet cu informații generale (data creării, data ultimei activări, relații de paternitate cu celelalte procese, drepturi de acces, relații de cooperare cu alte procese etc.), precum și date locale, specifice fiecărui proces în parte.

Nivelul aplicației asigură funcții ce sînt dependente de natura aplicației pentru care este conceput sistemul teleinformatic. Ca și nivelul sistemului de operare, el conține procese ierarhizate după modul cum furnizează servicii unele altora. Realizarea funcțiilor sistemului teleinformatic presupune cooperarea între procese, conform unor protocoale „croite" după specificul aplicației.

Firesc, sistemele teleinformatice au stimulat noi aplicații, cum sînt de exemplu cele telematice, destinate utilizării atît de către marele public, cît și de către specialiști. Amintim doar cîteva din aplicațiile de telematică, ale căror avantaje sînt deja evidente, cu toate că domeniul este abia la începutul dezvoltării sale.

Sistemele videotex permit vizualizarea pe ecrane de televizor obișnuit a informațiilor cerute de utilizatori. Informația este transmisă în formă codificată numeric, prin rețeaua de televiziune sau prin rețeaua telefonică, iar afișarea ei se realizează în formă alfanumerică și grafică. Informația este dispusă în pagini grupate într-o revistă (jurnal), la un moment dat putînd fi afișată pe ecran o pagină. Un terminal videotex cuprinde un receptor de televiziune, un decodificator și o claviatură care permit utilizatorului să aleagă revista și să o răsfoiască pagină cu pagină. Gradul de interactivitate dintre utilizator și centrul de servire este variabil și depinde de tipul sistemului. În unele cazuri utilizatorul poate transmite de la claviatura sa doar comenzi, recepționînd mesajele solicitate. În altele este posibil ca mesajele să fie transmise în ambele sensuri ale legăturii dintre utilizator și centrul de servire.

Sistemele teletex facilitează punerea la punct și transmiterea automată a corespondenței printr-o rețea de telecomunicații. Postul de lucru cuprinde claviatura, ecranul de afișare și imprimanta, grupate în jurul unui microprocesor dotat cu memorie externă pe disc flexibil. Textele introduse de la claviatură sînt afișate pe ecran în vederea verificării și editării. Totodată sînt memorate pe disc în vederea păstrării și transmiterii ulterioare prin rețeaua de comunicație. Copii permanente ale textelor pot fi obținute la imprimantă. Dactilografierea și transmiterea la distanță a textelor sînt completate de diferite servicii de autentificare a documentelor produse (datarea documentelor, identificarea corespondenților etc.).

Telecopierea constă în reproducerea la distanță a oricărui document, inclusiv fotografii, cu ajutorul unei rețele de telecomunicație. Pentru a transmite un document este necesar ca imaginea lui să fie codificată. O metodă simplă constă în baleierea documentului linie cu linie, cu un fascicul luminos reflectat pe o celulă fotosensibilă. Aceasta furnizează o succesiune de coduri reprezentînd intensitatea luminoasă a șirului de puncte ce alcătuiesc linia. Rezoluția uzuală este de 8 puncte de linie pe 1 mm și de aproximativ 4 linii pe 1 mm. Codurile obținute sînt transmise la distanță către un receptor, unde o imprimantă prin puncte (de exemplu, o imprimantă termică) redă documentul linie cu linie.

Telescrierea permite transmiterea desenelor trasate pe o tabletă specială și vizualizarea lor pe ecranul corespondentului. Sistemul de telescriere cuprinde stația de intrare, subsistemul de transport și dispozitivul de afișare. Stația de intrare este prevăzută — pe lîngă tableta de trasare — cu o claviatură pentru introducerea comenzilor, între care un loc aparte îl ocupă comenzile de stabilire a modului de trasare: trasarea continuă, ștergerea, schimbarea culorii, poziționarea cursorului etc.

Desenele trasate pe tabletă cu un crelon special sînt reprezentate prin puncte și sînt codificate ca o listă de perechi de valori, fiecare pereche indicînd coordonatele unei anumite poziții atinse de vîrful creionului pe tabletă. Pentru transmiterea prin subrețeaua de transport datele pot fi comprimate, succesiunile de puncte care au aceeași abscisă sau ordonată fiind recodificate ca vectori cu o anumită lungime. Desenele sînt reproduse pe ecranul corespondentului tot prin puncte, după ce — bineînțeles — datele au fost adaptate la modul de afișare prin baleiaj, caracteristic majorității dispozitivelor de vizualizare.

Teleconferința sau ținerea de reuniuni la distanță permite mai multor grupuri aflate în locuri diferite unele de altele să converseze, să schimbe documente prin telecopiere ori să utilizeze sistemul de telescriere. Fiecare participant are la dispoziție un panou de avertizare, care permite sesizarea intervenției interlocutorilor și identificarea lor.

Vizioconferința folosește serviciul telefonic completat cu cel de transmitere a imaginii corespondentului. Terminalul viziofon încorporează un receptor de imagini, o cameră de luat vederi, un microfon și un pupitru de comandă. Pupitrul de comandă permite controlul cadrajului imaginii și comutarea ei pentru vizualizarea unor documente. Imaginea, codificată numeric, este transmisă pe un canal cu debitul de 2 Mbiți pe secundă.

Birotica constă în transpunerea documentelor sub formă de text, sunet sau imagine, în comunicarea lor, ca și în serviciile de mesagerie, poștă electronică, curier electronic, agendă electronică etc. Documentele sînt prelucrate la posturile electronice de lucru, care cuprind mijloace diverse de introducere a informației: claviatură, microfon, cameră de luat vederi, precum și un ecran de vizualizare. Informația poate fi transmisă prin rețele locale, la nivelul unor întreprinderi, sau prin rețele cu sateliți pentru distanțe mari.

Caracteristicile sistemelor teleinformatice se vor regăsi desigur la viitoarele sisteme de calcul din generația a cincea. Conform proiectului, sistemele experte, mai precis cele de prelucrare a cunoștințelor vor reprezenta principalele aplicații ale calculatoarelor în anii '90. Realizarea lor va face apel la prelucrări distribuite sau paralele, caracteristice unor rețele de calculatoare în alcătuirea cărora vor intra diferite clase de componente, de la supercalculatoare — distribuite geografic —, la mini și microcalculatoare pe o singură pastilă. Arhitectura sistemelor de calcul din generația a cincea nu va mai fi cea actuală, ci una care să exploateze cît mai bine paralelismul inerent aplicațiilor din categoria sistemelor experte. De exemplu, o arhitectură paralelă de tip flux de date, în care execuția unei prelucrări este declanșată cînd datele de intrare necesare devin disponibile, este mai potrivită pentru viitoarele sisteme de calcul. Oricum, experiența cîștigată prin realizarea actualelor sisteme teleinformatice va influența într-o măsură importantă tehnologia specifică elaborării calculatoarelor din generația a cincea.

1.3. INFLUENȚA DEZVOLTĂRII TELECOMUNICAȚIILOR ASUPRA ARHITECTURII SISTEMELOR TELEINFORMATICE

Subrețelele de comunicație bazate pe telecomunicații convenționale, prezentate în detaliu în [P-1], sînt supuse unor modificări și perfecționări continue impuse de: diversificarea terminalelor utilizatorilor; asigurarea con-

Tratarea informației / Mediu pentru informație		Procesarea informației (în sens strict)	Depozitarea și regăsirea informației	Transferul informației
Voce		Calculatoare în accepțiune originală și convețională (Procesare dată/informație)		Comunicații convețională
Dată				
Text				
Imagine fixă	Desene (imagini codate)			
	Desene de tip scanning (inclusiv foto)			
Imagine în mișcare	Desene (inclusiv animație)			
	Desene de tip scanning (inclusiv foto)			

COMUNICAȚII MODERNE

(după [K-1])

Fig. 1.5. Relația dintre comunicațiile moderne și cele convenționale

trolului comunicațiilor atît de către partea apelantă, cît și de cea apelată; reducerea dificultăților în utilizare; solicitarea tot mai mare a echipamentelor de tip recopietoare de ecran; dezvoltarea bazelor de date; creșterea volumului datelor care fac obiectul comunicațiilor.

(Tele)comunicațiile moderne reușesc să integreze transferul informației cu depozitarea și regăsirea informației, precum și cu procesarea (în sens strict) a ei [K-1]. În figura 1.5. este prezentată relația dintre comunicațiile moderne și cele convenționale. Împărțirea tratării informației în cele trei părți (fig. 1.5) corespunde scopului funcțiilor de bază ale comunicațiilor moderne (de transfer, depozitare și procesare). În figura 1.6. sînt ilustrate legăturile dintre aceste funcții.

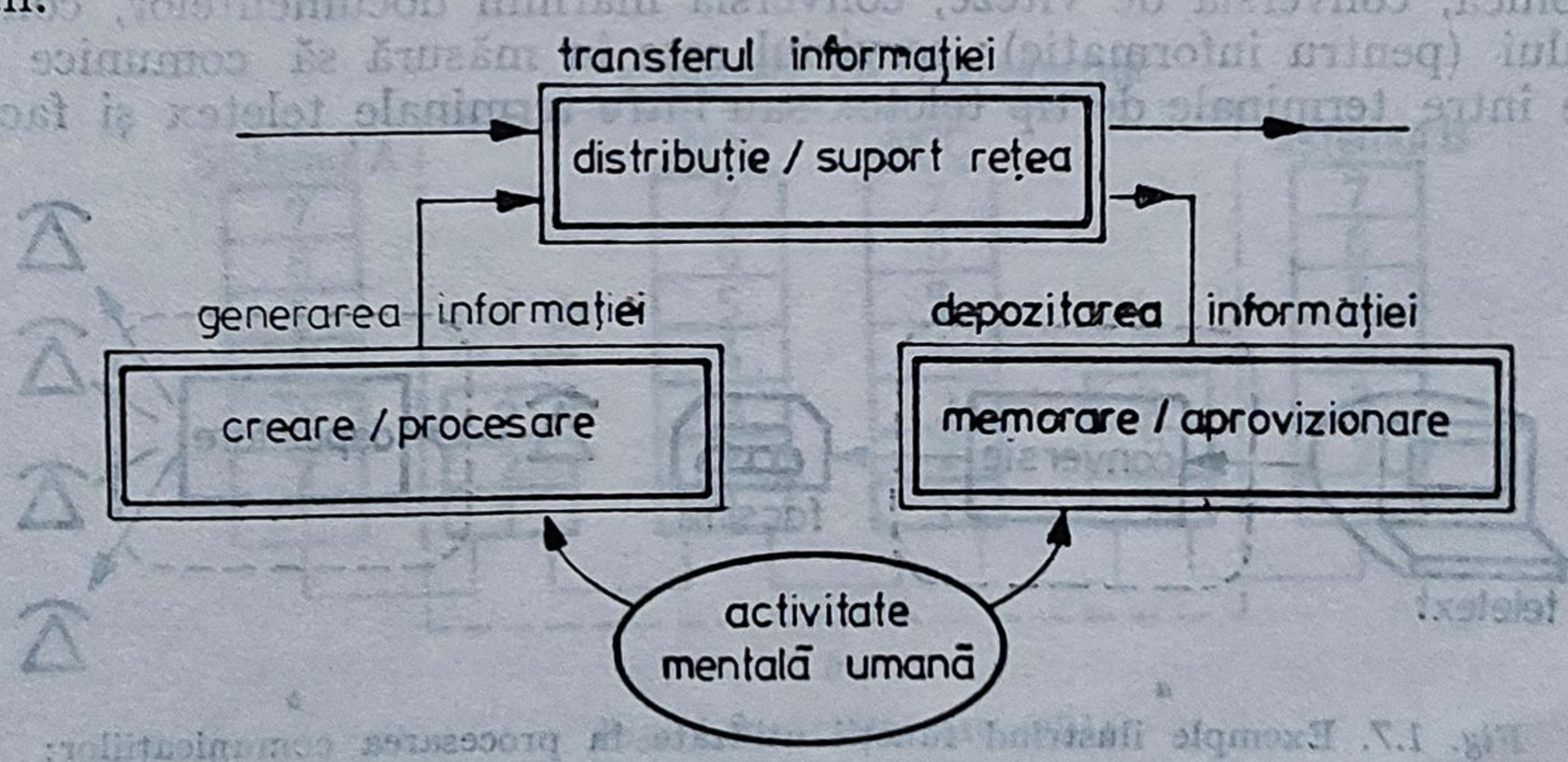


Fig. 1.6. Legături între funcțiile de bază ale comunicațiilor moderne

Realizarea unei subrețele moderne de (tele)comunicații în care informația transmisă în cantitate mare să poată fi depozitată și procesată presupune — în primul rînd — înlocuirea sistemelor prezente de comutare (necesare pentru telefonia analogică) cu sisteme de comutare digitale, capabile să controleze și echipamentele digitale ale utilizatorului. În finalul construcției, asemenea subrețea va fi în întregime digitizată. Digitizarea subrețelelor de comunicație atrage după sine dezvoltări noi atît ale funcțiilor de comutare, cît și ale celor de procesare din fiecare nod al subrețelei; concepte noi, mai cuprinzătoare decît — de exemplu — cel al ISDN-ului (Integrated Services Digital Network) vin să susțină aceste lucruri.

Conceptul INS (Information Network System) [K-2] presupune dezvoltarea funcțiilor subrețelei de comunicație pentru a oferi posibilități noi și diversificate de comunicație între calculatoare sau între calculatoare și terminale folosind mediul vocal, de date ori imagine, toate acestea disponibile la un preț de cost convenabil. Pentru materializarea conceptului INS și realizarea unei rețele moderne de comunicații unele țări și-au creat proiecte-pilot [T-1], prin care urmăresc confirmarea tehnică a unei rețele în întregime digitizate la care sînt conectate echipamentele beneficiarului; în plus se mai are în vedere și obținerea de informații tehnice și păreri referitoare la serviciile deja furnizate de o asemenea rețea, modele de utilizare a serviciilor, precum și evidențierea cerințelor pentru dezvoltarea unor servicii noi.

Diversificarea comunicațiilor se poate realiza prin oferirea de către subrețeaua de comunicații a unor servicii noi, care implică și procesarea comunicațiilor. Serviciile de procesare a comunicațiilor urmăresc mai multe obiective, între ele numărîndu-se: realizarea comunicațiilor între terminale de tipuri foarte diferite; stabilirea unor metode noi de comunicație; suportarea procesării informației; asigurarea unui cost redus pentru comunicații. Aceste servicii utilizează funcții de conversie (de exemplu, a mediului de comunicație — fig. 1.7a— sau a formatului) și funcții de depozitare a informațiilor (fig. 1.7b). Dintre serviciile care presupun procesarea comunicațiilor în conceptul INS menționăm pe cel:

- *de memorare a vocii*, bazat pe posibilitatea (realizată prin folosirea funcției de memorare) de comunicare a vocii; serviciul poate livra la o destinație anumită mesaje vocale, buletine informative, anunțuri;
- *de comunicare a mesajelor*, bazat pe transportul mesajelor, poșta electronică, conversia de viteze, conversia mărimii documentelor, conversia mediului (pentru informație); serviciul este în măsură să comunice documente între terminale de tip teletex sau între terminale teletex și facsimil;

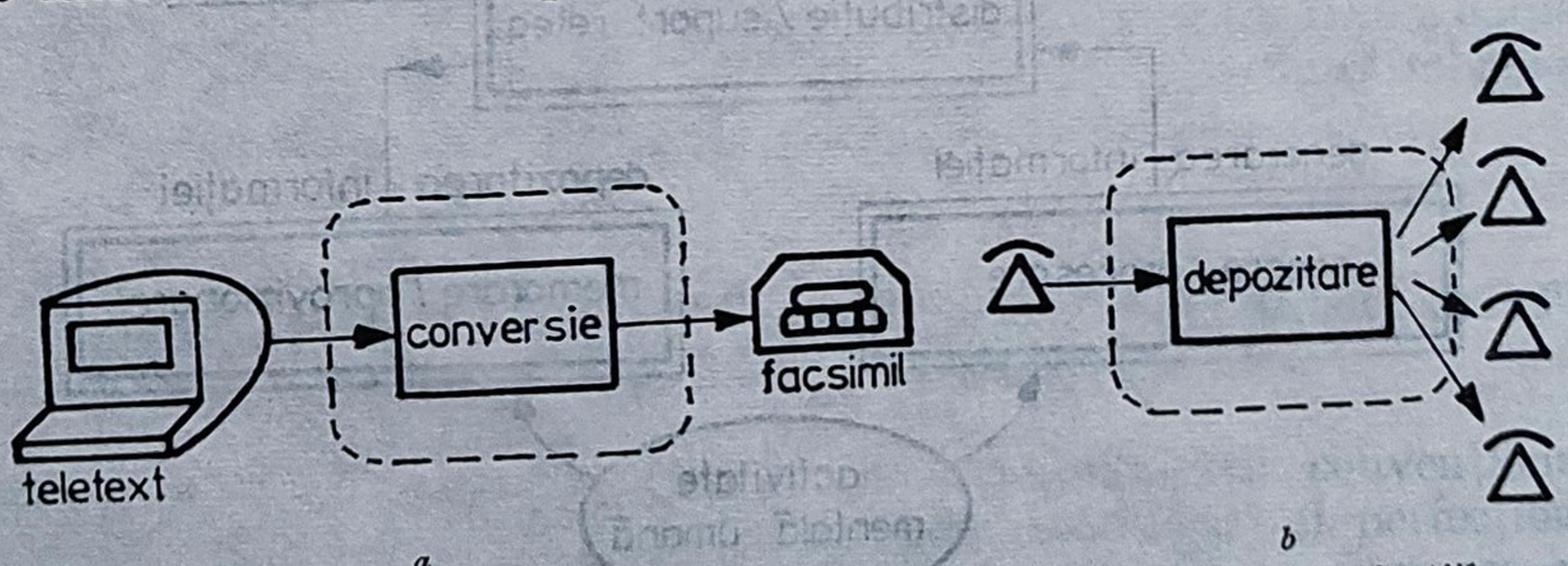


Fig. 1.7. Exemple ilustrînd funcții utilizate în procesarea comunicațiilor: *a* — conversie de mediu; *b* — depozitare a informațiilor

• *pentru comunicații de tip facsimil-digital,* beneficiind de livrări la adrese multiple, de poștă facsimil (livrările de mesaje confidențiale, regăsirea unui mesaj provenit de la *N* emitori) și conversii ale mediului informației, realizează comunicații facsimil între terminale facsimil și centrele de informare;

• *pentru comunicațiile digitale ale imaginilor fixe* (de exemplu, pentru videotext);

• *de acces la o bază de date pornind de la medii multiple;* informația cu care se lucrează poate fi sub formă alfanumerică, de imagine sau voce, deoarece se pot asigura medii multiple și pentru ieșire.

Măsura în care perfecționările în telecomunicații acționează asupra arhitecturii sistemelor teleinformatice va fi ilustrată prin exemplul următor. Fie două sisteme de calcul *A* și *B*, numite și sisteme-terminale, care — spre a putea comunica între ele — folosesc o subrețea de comunicații. Din punctul de vedere al modelului de referință pentru interconectarea sistemelor deschise (OSI/ISO) subrețelele de comunicație convențională apar așa cum sînt ele ilustrate în figura 1.8a. În acest caz informația utilizatorului este transportată de la sistemul *A* la sistemul *B* transparent, folosind totodată tehnici multiple de protecție la erorile ce pot apărea în timpul transportului. Informația transportată urmează să fie utilizată de niveluri superioare nivelului *3* din sistemele terminale. În subrețelele de comunicații moderne nodurile rețelei care participă la asigurarea serviciilor utilizează nivelurile superioare nivelului 3, în scopul realizării funcțiilor de procesare a comunicațiilor. Aceste noduri, numite noduri de procesare a comunicațiilor (NPC) produc modificarea arhitecturii sistemelor teleinformatice (fig. 1.8b).

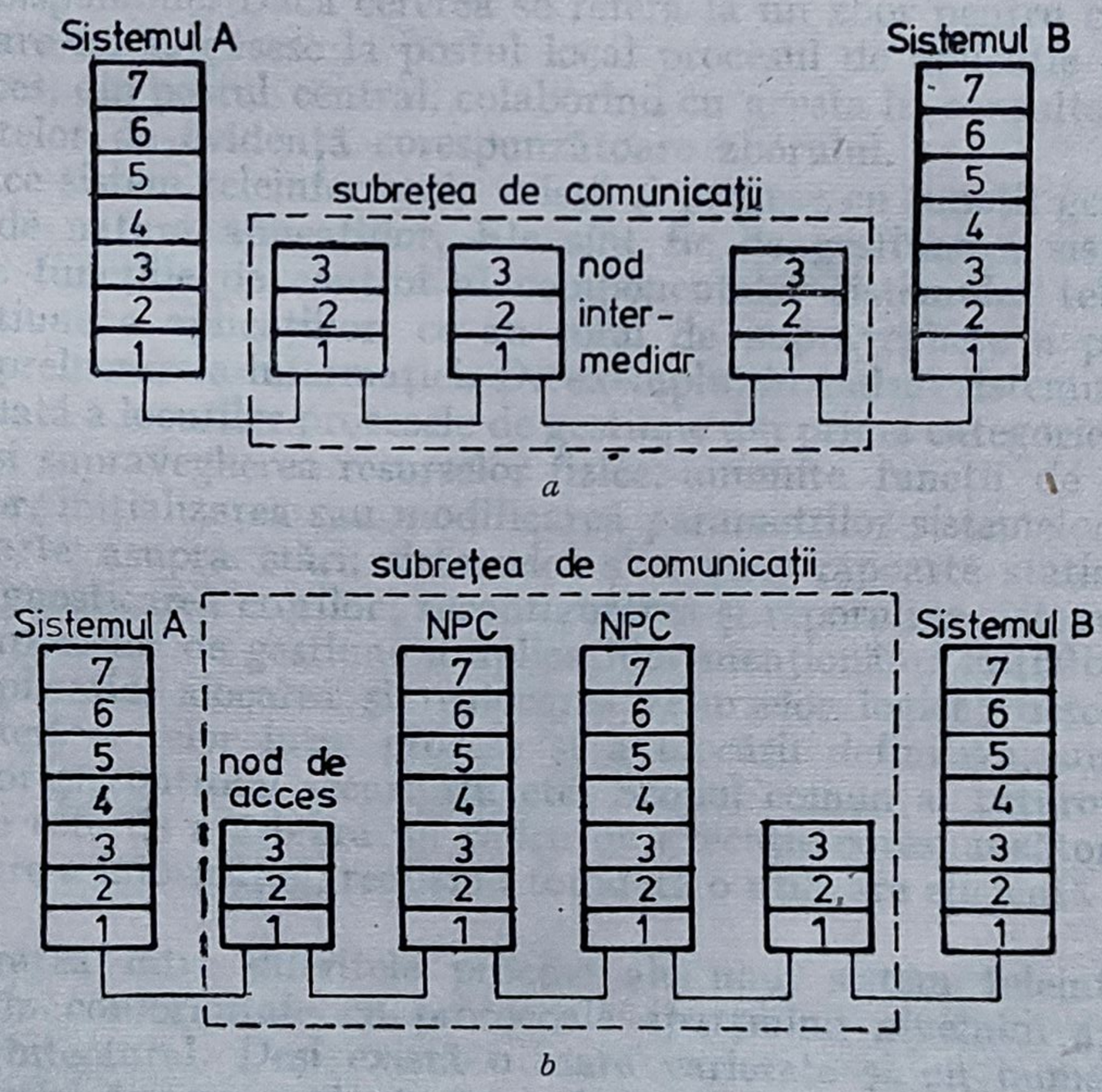


Fig. 1.8. Arhitectura OSI:
a — clasică; *b* — modificată în baza conceptului INS

Dacă ținem seama de conceptul INS, în ansamblul rețelei teleinformatice bazată pe comunicații moderne apar trei activități importante:

• *activitatea de transport* care are ca obiect mișcarea informației utilizatorului, informație asupra căreia însă nu se intervine sub nici o formă. Această activitate poate fi realizată de o rețea cu comutare de circuite, o rețea cu comutare de pachete.

• *activitatea de procesare a comunicațiilor* care presupune lucru asupra informației utilizatorului (de exemplu prin conversii ale formatului datelor, conversii de protocoale), dar care nu afectează semnificația informației utilizatorului.

• *activitatea de procesare a informației* care presupune procesarea informațiilor pentru a obține datele obiectiv.

Nevoile utilizatorilor și tehnicile cerute pentru a putea realiza servicii de procesare a comunicațiilor sînt astăzi în faza unor căutări permanente. Transpunerea în viață a conceptului INS trebuie considerată ca o perspectivă pe termen lung, poate în primul rînd datorită echipamentelor analogice existente în telecomunicațiile din toate țările, care vor fi anevoios înlocuite.

CAPITOLUL 2

UTILIZAREA MODELELOR DE CALCUL DISTRIBUIT ÎN PROIECTAREA SISTEMELOR TELEINFORMATICE

Aplicațiile caracteristice sistemelor teleinformatice sînt de natură distribuită. Obiectivele unei astfel de aplicații nu pot fi atinse prin execuția unui singur proces, ci prin acțiunile corelate ale mai multor procese, care trebuie să comunice între ele în vederea cooperării. De exemplu, să considerăm un sistem de rezervare automată a locurilor de avion, realizat pe baza unei rețele de calculatoare cu posturi locale de prelucrare, în diverse agenții de voiaj și un post central de prelucrare. Pentru fiecare cerere de rezervare angajatul agenției, a cărui activitate poate fi asimilată unui proces din sistemul teleinformatic, declanșează un dialog cu un proces de aplicație, care are ca funcții consultarea și actualizarea listelor de evidență a zborurilor, a locurilor ocupate și a celor disponibile. Dacă cererea se referă la un zbor pentru care informațiile necesare nu se găsesc la postul local procesul de aplicație comunică cu un alt proces, din postul central, colaborînd cu acesta în consultarea și actualizarea datelor de evidență corespunzătoare zborului.

În orice sistem teleinformatic există și procese cu funcții generale, independente de natura aplicațiilor. Ele sînt fie de gestiune a sistemelor, care îndeplinesc funcțiile de control al componentelor sistemului teleinformatic, fie de gestiune a aplicațiilor, ce au rolul de supraveghere a proceselor cu sarcini de prelucrare a informației. De exemplu, în cadrul sistemului de rezervare automată a locurilor procesele de gestiune din prima categorie pot realiza: activarea și supravegherea resurselor fizice, anumite funcții de încărcare a programelor, inițializarea sau modificarea parametrilor sistemelor; alcătuirea unor rapoarte asupra stării sistemelor și a unor rapoarte statistice; detectarea și diagnosticarea erorilor; reconfigurarea și repornirea sistemelor. Dintre funcțiile proceselor de gestiune a aplicațiilor menționăm: inițializarea proceselor de aplicație, alocarea și realocarea resurselor logice; detecția și prevenirea interferențelor între procese și a blocării definitive; autentificarea utilizatorilor și controlul securității etc. Scopul comun al tuturor proceselor de gestiune este de a asigura un cadru de execuție corespunzător proceselor de prelucrare a informației, realizînd totodată o utilizare eficientă a resurselor de calcul.

Cooperarea între diferitele procese ale unui sistem teleinformatic se realizează în conformitate cu protocoale aparținînd nivelului aplicație din modelul arhitectural. Deși există o mare varietate și un număr potențial imens de astfel de protocoale, toate au la bază o serie de mecanisme comune de comunicare și de sincronizare între procese.

2.1. MODELE ALGORITMICE DE CALCUL PARALEL

Analiza oricărui sistem complex este facilitată de descompunerea sa în părți mici și simple. Sistemul poate fi astfel înțeles mai bine, prin cunoașterea diferitelor componente și a relațiilor dintre ele. Adoptarea acestei idei la nivelul specific al sistemelor de calcul a generat o serie de preocupări vizînd stabilirea unor unități și procedee de descompunere cît mai convenabile. Procesul secvențial la care ne vom referi în continuare constituie o astfel de unitate de reprezentare a comportării unui sistem de calcul. Elemente primitive vor fi considerate datele și operațiile aplicate datelor. Intuitiv, datele constituie reprezentări ale informației în calculator, iar operațiile sînt regulile de transformare a datelor. Execuția unei operații constituie un eveniment. În funcție de aplicație, un eveniment poate fi execuția unei proceduri sau a unei singure instrucțiuni-mașină. Rezultatul operației nu depinde de viteza cu care ea este executată. Prin calcul se înțelege mulțimea finită de operații aplicate unei mulțimi finite de date în scopul rezolvării unei probleme. Operațiile calculului trebuie executate într-o ordine determinată, pentru ca rezultatele anumitor operații să poată fi utilizate de altele. Cea mai simplă posibilitate de ordonare este cea strict secvențială, în care operațiile se efectuează cîte una la un moment dat. Avem de-a face, în acest caz, cu un proces secvențial, care constă dintr-o mulțime de evenimente total ordonate în timp. Să luăm ca exemplu problema evaluării expresiei aritmetice $y = ((a + b + c)/(d + e + f) - g) \times h$. O posibilitate de ordonare strictă a operațiilor prin care valoarea expresiei poate fi calculată este ilustrată de graful de precedență din figura 2.1 în care cercurile reprezintă diferitele operații, iar arcele orientate, ordinea operațiilor. Se observă însă că evaluarea expresiei date cere doar o ordonare parțială a operațiilor în timp, cum este cea ilustrată de graful de precedență din figura 2.2.

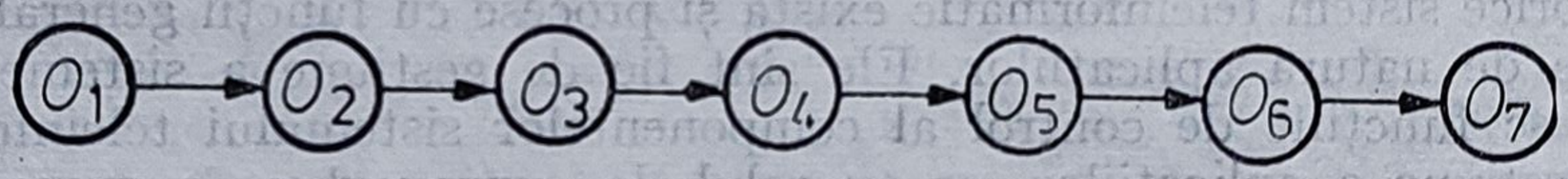


Fig. 2.1. Evaluarea unei expresii aritmetice:
$O_1 - t_1 := a + b$; $O_2 - t_2 := t_1 + c$; $O_3 - t_3 := d + e$; $O_4 - t_4 := t_3 + f$;
$O_5 - t_5 := t_2/t_4$; $O_6 - t_6 := t_5 - g$; $O_7 - y := t_6 \times h$.

Ordonarea parțială permite execuția în paralel a unor operații. Astfel, pentru exemplul dat O_1 se poate executa în paralel cu O_3 și O_4; de asemenea, O_2 cu O_3 și O_4. Desigur, exemplul este simplu și nu are o importanță practică deosebită, însă permite cunoașterea aspectelor ce caracterizează sistemele de calcul. Astfel, înțelegerea funcționării acestor sisteme poate fi facilitată de descompunerea calculelor paralele într-un număr de procese secvențiale care se analizează separat. Pentru exemplul dat există posibilitatea grupării

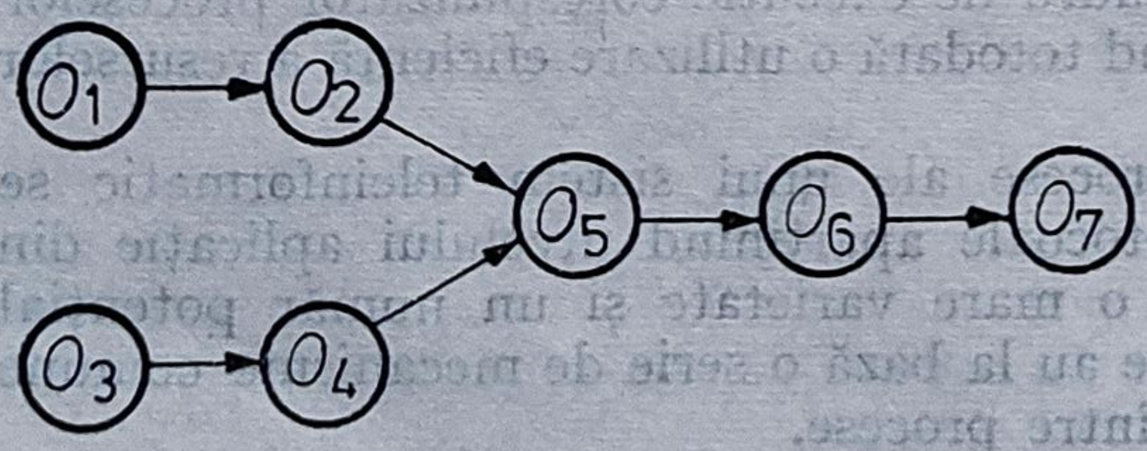


Fig. 2.2. Ordonarea parțială a operațiilor la evaluarea unei expresii.

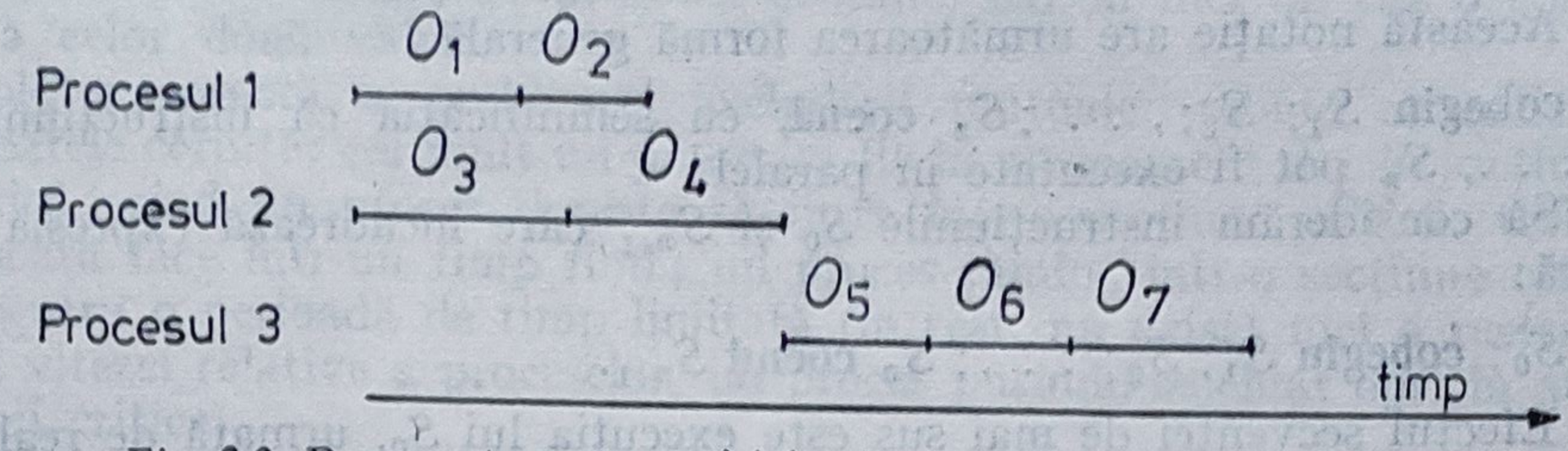


Fig. 2.3. Reprezentarea execuției în timp a trei procese paralele.

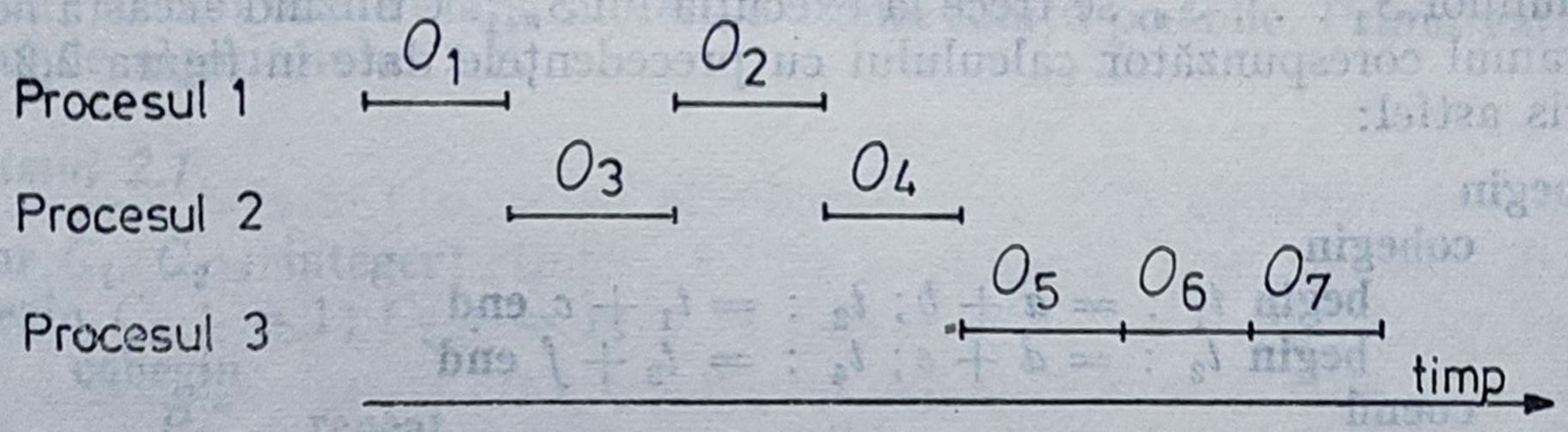


Fig. 2.4. O altă execuție în timp a trei procese paralele

calculelor în trei procese secvențiale: unul ce cuprinde operațiile O_1 și O_2, al doilea care include operațiile O_3 și O_4 și al treilea conținînd operațiile O_5, O_6 și O_7.

Descompunerea în procese secvențiale a unui sistem induce, de cele mai multe ori, o clasificare a datelor fiecărui proces în date locale, care intervin doar în operațiile procesului respectiv, și în date comune, ce sînt utilizate și în alte procese. Astfel, a, b și c sînt date locale procesului 1, în timp ce t_2 (fig. 2.1) este dată comună proceselor 1 și 3.

Procesele secvențiale interacționează prin legăturile realizate de datele comune. Aceste legături sînt slabe (procesele sînt slab conectate), deoarece procesele — exceptînd momentele de interacțiune, ca cele din exemplul anterior — pot evolua cu viteze diferite. Singurul element important este ca operația O_5 a procesului 3 să fie efectuată după ce procesele 1 și 2 le-au calculat pe t_2, respectiv pe t_4.

Două procese sînt paralele sau concurente dacă execuțiile lor se suprapun în timp (fig. 2.3). Ele se consideră paralele (concurente) chiar dacă operațiile lor nu se suprapun, dar sînt efectuate intercalat în timp (fig. 2.4). Sînt, evident, posibile și situații în care anumite operații ale proceselor se suprapun, iar altele se intercalează în timp. Oricum, în situația că prima operație a unui proces începe înainte de terminarea ultimei operații a altuia cele două procese sînt paralele.

2.1.1. SECȚIUNI CRITICE

Descrierea riguroasă a calculelor în vederea realizării lor pe calculator impune utilizarea unui limbaj de programare. În cazul calculelor paralele limbajul de programare trebuie să conțină notații care să permită exprimarea posibilității execuției în paralel a unor instrucțiuni. O asemenea notație a fost propusă de Dijkstra [D-1.], fiind preluată ulterior și de alți autori [H-1.].

Această notație are următoarea formă generală:

cobegin S_1; S_2; . . . ; S_n **coend,** cu semnificația că instrucțiunile S_1, S_2, . . ., S_n pot fi executate în paralel.

Să considerăm instrucțiunile S_0 și S_{n+1} care încadrează expresia anterioară:

S_0; **cobegin** S_1; S_2; . . .; S_n **coend** S_{n+1}.

Efectul secvenței de mai sus este execuția lui S_0, urmată de realizarea în paralel a instrucțiunilor S_1 . . . S_n. Cînd încetează execuția tuturor instrucțiunilor S_1 . . . S_n, se trece la execuția lui S_{n+1}. Utilizînd această notație, programul corespunzător calculului cu precedențele date în figura 2.2 poate fi scris astfel:

```
begin
    cobegin
        begin t1 := a + b; t2 := t1 + c end
        begin t3 := d + e; t4 := t3 + f end
    coend
    t5 := t2/t4; t6 := t5 − g; y := t6 × h
end
```

Folosirea notației anterioare reclamă puțină atenție. Să considerăm că S_i și S_j sînt două procese paralele și că S_i utilizează o variabilă pe care o modifică S_j. Evident, rezultatul execuției lui S_i poate depinde de viteza lui S_j (deci de momentul la care S_j modifică variabila). Asemenea comportări dependente de timp (care pot genera erori) sînt excluse dacă instrucțiunile S_1, S_2, . . ., S_n, din notația

cobegin S_1; S_2; . . .; S_n **coend**

descriu procese disjuncte, adică procese ce operează pe mulțimi de variabile disjuncte. Același lucru se întîmplă cînd procesele utilizează variabile comune pe care nici unul din ele nu le modifică.

În cazul general, al proceselor paralele care au acces necondiționat la variabile comune, sînt necesare măsuri suplimentare de sincronizare. Prin sincronizare se înțelege orice constrîngere asupra ordonării operațiilor în timp. O regulă de sincronizare larg utilizată este excluderea mutuală a operațiilor în timp, ce poate fi exprimată ca o restricție de forma: operațiile O_i și O_j să nu se execute în același timp. O posibilitate de obținere a acestui efect este cea la care ne referim în continuare. Să considerăm două procese secvențiale, P_1 și P_2, avînd o comportare ciclică. În fiecare ciclu putem identifica o secțiune, pe care o numim secțiune critică, avînd proprietatea că în orice moment cel mult unul dintre cele două procese poate fi angajat în secțiunea sa critică. Pentru a realiza această excludere mutuală procesele au acces la un grup de variabile comune. În continuare se consideră că inspectarea valorii curente și modificarea valorii unei variabile comune sînt acțiuni indivizibile, neinterferabile. Cu alte cuvinte, cînd cele două procese încearcă să execute simultan două acțiuni asupra aceleiași variabile comune rezultatul este identic cu cel al efectuării acțiunilor una după alta[1]. Astfel,

---

[1] Acest efect este realizat, de exemplu, de mecanismul de arbitrare al memoriei variabilelor.

inspectarea unei variabile comune simultan cu modificarea ei va da fie vechea valoare, fie noua valoare a variabilei comune, dar în nici un caz o combinație a celor două valori.

Soluția corectă a problemei excluderii mutuale trebuie să satisfacă următoarele cerințe: cel mult un proces să fie la un moment dat în secțiunea sa critică; cînd un proces dorește să intre într-o secțiune critică, trebuie să o poată face într-un timp finit; un proces rămîne într-o secțiune critică doar pentru o perioadă de timp limitată (în rest, nu există nici o restricție asupra vitezei relative a proceselor, un proces putîndu-se chiar opri în afara secțiunii critice).

Pentru a înțelege mecanismele secțiunilor critice prezentăm modul de construire în mai multe etape succesive a unei soluții posibile. Prima variantă corespunde algoritmului 2.1.

Algoritmul 2.1

```
var C1, C2 : integer;
begin C1 := 1; C2 := 1;
    cobegin
„P1“ : repeat
           C1 := 0;
           while C2 = 0 do (* nimic *);
           * secțiune critică 1;
           C1 := 1;
           * secțiune rest 1
       forever
„P2“ : repeat
           C2 := 0;
           while C1 = 0 do (* nimic *);
           * secțiune critică 2;
           C2 := 1;
           * secțiune rest 2
       forever
    coend
end
```

Descrierea algoritmului utilizează notațiile limbajului PASCAL, căruia i s-au adăugat construcțiile sintactice specifice acțiunilor paralele (**cobegin** . . . **coend**) și repetării nelimitate a unei secvențe de operații (**repeat** . . . **forever**). Soluția folosește două variabile comune C_1 și C_2, avînd valori întregi, 0 sau 1. Variabila C_1 este modificată doar de procesul P_1, luînd valoarea 0 înainte de intrarea procesului în secțiunea critică și valoarea 1 la ieșirea din ea. C_1 poate fi inspectată de procesul P_2, indicînd acestuia dacă P_1 se află în interiorul ori în afara secțiunii sale critice. Un rol similar îl joacă variabila C_2 în procesul P_2. Pentru a avea convingerea că algoritmul asigură excluderea mutuală, este necesar să fie urmărit momentul în care procesul P_1, după ce găsește că $C_2 = 1$, decide să intre în secțiunea critică 1. În această situație C_1 este zero și va continua să rămînă la o astfel de valoare pînă cînd P_1 iese din secțiunea critică 1. Deoarece $C_2 = 1$, procesul P_2 se află înaintea secțiunii critice 2, în care nu poate intra cît timp C_1 rămîne zero. Din nefericire, soluția prezintă pericolul blocării definitive a proceselor. În

cazul în care, procesul P_1 dă lui C_1 valoarea 0 și — înainte ca el să testeze C_2 (**while** $C_2 = 0$ . . .) — P_2 dă lui C_2 valoarea 0 fiecare din cele două procese rămîne definitiv în ciclul **while** dinaintea secțiunii critice corespunzătoare. Evitarea acestei situații este posibilă prin înlocuirea ciclurilor **while** inefective cu unele în care variabilele C_1 și C_2 iau — alternativ — valorile 1 și 0. Pentru procesul P_1 se poate scrie:

```
. . . . . .
while C2 = 0 do
    begin C1 : = 1;
          C1 : = 0
    end;
. . . . . .
```

Astfel, variabila C_1 revine periodic la valoarea 1, creînd procesului P_2 posibilitatea să treacă în secțiunea critică. Soluția fiind simetrică, sînt aduse în discuție și variabila C_2 și procesul P_1. Nici această soluție nu este acceptabilă, deoarece ea nu este total independentă de vitezele relative ale celor două procese. Ca atare este posibil ca pentru anumite viteze fiecare proces (P_1 sau P_2) să inspecteze variabila (C_2 sau C_1) doar atunci cînd are valoarea zero, procesele blocîndu-se definitiv. În cazul unui conflict între procese la intrarea în secțiunea critică soluția poate fi corectată prin impunerea unuia dintre ele să cedeze, menținînd variabila sa C pe 1 pînă cînd celălalt execută secțiunea critică. Deoarece nici unul dintre procese nu trebuie favorizat, cel care urmează să intre în secțiunea critică (în cazul conflictului) este indicat de o variabilă — *turn*, care ia alternativ valorile 1 și 2. Soluția este prezentată în algoritmul 2.2.

Algoritmul 2.2.

```
var C1, C2, turn: interger;
begin C1 : =1, C2 : = 1; turn : = 1;
    cobegin
„P1“ : repeat
           C1 : = 0;
           while C2 = 0 do
               if turn = 2 then
                   begin
                       C1 : = 1;
                       repeat until turn = 1;
                       C1 : = 0
                   end;
           * secțiune critică 1;
           turn : = 2; C1 : = 1;
           * secțiune rest 1
       forever
```

```
„P2“ : repeat
          C2 : = 0;
          while C1 = 0 do
              if turn = 1 then
                  begin
                      C2 : = 1;
                      reapeat until turn = 2;
                      C2 : = 0
                  end;
              * secțiune critică 2;
                turn : = 1; C2 : = 1;
              * secțiune rest 2
          forever
      coend
end
```

Ca și în variantele anterioare procesul P_1 testează C_2 numai cînd $C_1 = 0$ și intră în secțiunea critică doar dacă $C_2 = 1$. Deoarece aceeași observație este valabilă pentru procesul P_2, rezultă că excluderea mutuală este asigurată.

Procesele P_1 și P_2 nu se blochează definitiv înaintea secțiunii critice. Variabila *turn* se modifică doar la ieșirea dintr-o secțiune critică, rămînînd constantă cînd ambele procese se află înaintea secțiunii critice. Dacă $turn = 1$ cînd ambele procese încearcă să intre în secțiunea critică, atunci P_2 va face atribuirea $C_2 : = 1$ și va aștepta pînă ce *turn* va deveni egal cu 2. În felul acesta procesul P_1 va putea pătrunde în secțiunea critică. Un raționament similar poate fi făcut pentru cazul în care $turn = 2$. În fine, oprirea procesului P_2 în secțiunea rest nu împiedică execuția secțiunii critice de către P_1, deoarece C_2 are valoarea 1. Condiția simetrică este și ea îndeplinită.

Soluția prezentată admite o generalizare imediată pentru cazul excluderii mutuale a N procese. Dar atît soluția de bază cît și generalizarea ei conțin acțiuni de testare și modificare a unei variabile *(turn)* de către toate procesele. De aceea ele devin neoperaționale în cazul defectării memoriei care conține această variabilă comună. O variantă ce elimină neajunsul [L-2.] permite ca fiecare proces să fie executat pe un procesor separat, avînd o memorie proprie. Un procesor poate citi din memoria altuia, dar poate scrie doar în memoria proprie. Algoritmul este robust, el funcționînd chiar dacă se defectează oricare dintre procesoare. Se presupune că în cazul unei defectări procesul corespunzător trece imediat în secțiunea rest și se oprește. Procesul poate fi repornit ulterior, din aceeași secțiune. Soluția permite intrarea proceselor în secțiunea critică în ordinea în care și-au anunțat intenția (primul venit — primul servit). Algoritmul este asemănător celui de servire a clienților într-un magazin. Fiecare client primește un număr de ordine la intrarea în magazin, posesorul celui mai mic număr fiind următorul servit.

Variabilele comune sînt două tablouri de cîte N elemente (N fiind numărul proceselor): *alege* și *nr*. Elementele *alege* [i] și *nr* [i] se află în memoria procesorului i și sînt inițial zero. Elementul *alege* [i] ia valoarea 1 doar în perioada cînd procesul i alege un număr de ordine; *nr* [i] indică numărul de ordine ales de procesul i. La rîndul lor, procesele sînt notate de la *1* la N. Algoritmul prevede că două procese care au același număr de ordine intră în secțiunea critică în ordinea dată de notația lor.

Soluția pentru procesul i este prezentată în algoritmul 2.3. Execuția trebuie să înceapă în afara secțiunii critice. Menționăm că funcția „maxim" dă ca rezultat cea mai mare valoare a argumentelor. De asemenea, relația $(a, b) < (c, d)$ este adevărată dacă $a < c$ sau dacă $a = c$ și $b < d$.

Algoritmul 2.3.

```
.....
„P" : repeat
        alege [i] : = 1;
        nr [i] : = 1 + maxim (nr [1], nr [2],. . ., nr [N]);
        alege [i] : = 0;
        for j : = 1 to N do
          begin
            L1 : repeat until alege [j] = 0;
            L2 : repeat until nr [j] = 0 ∨ (nr [j], j) < (nr[i], i)
          end;
        * secțiune critică i;
        nr [i] : = 0;
        * secțiune rest i;
      forever
.....
```

Convenim să spunem că procesul P_i este *în intrare* cît timp *alege* $[i] = 1$ și că este *în magazin* din momentul în care *alege* $[i] = 0$ pînă cînd părăsește secțiunea critică.

Pentru a arăta că soluția este corectă observăm mai întîi că dacă procesele i și k sînt *în magazin* și i a ajuns *în magazin* înainte ca procesul k să ajungă *în intrare*, atunci procesul i a făcut alegerea înaintea lui k și $nr\ [i] < nr\ [k]$.

În situația că procesul i este în secțiunea sa critică, iar procesul k (diferit de i) este *în magazin*, atunci $(nr\ [i], i) < (nr\ [k], k)$. În scopul probării celor afirmate se iau în discuție timpii de producere ai unor evenimente, considerați din punctul de vedere al procesului P_i. Fie t_{L1} timpul la care P_i găsește *alege* $[k]$ egal cu 0 la ultima parcurgere a ciclului L_1 și t_{L2} timpul cînd procesul i începe ultima execuție a lui L_2 pentru $j = k$. Evident $t_{L1} < t_{L2}$. De asemenea, fie t_i timpul la care procesul k pătrunde *în intrare*, t_s — timpul cînd a terminat scrierea valorii lui $nr\ [k]$, iar t_e — timpul la care a părăsit intrarea. Evident, $t_i < t_s < t_e$. Deoarece *alege* $[k] = 0$ la timpul t_{L1}, găsim că (a) fie $t_{L2} < t_i$, (b) fie $t_e < t_{L2}$. În cazul (a), datorită observației anterioare, $nr\ [i] < nr\ [k]$. În situația (b), $t_s < t_e < t_{L2}$, deci $t_s < t_{L2}$. Așadar procesul i a găsit în $nr\ [k]$ valoarea curentă (actualizată la t_s) și — deoarece a ajuns în secțiunea critică — înseamnă că $(\text{nr}\ [i], i) < (nr\ [k], k)$.

În cazul că nici un proces nu este în secțiunea critică și există mai multe procese *în magazin*, atunci cel cu $(nr\ [i], i)$ minim termină execuția ciclului **for** și intră în secțiunea critică.

În concluzie, rezultă că în secțiunea critică se poate găsi cel mult un proces, procesele intră în secțiunea critică în ordinea primul venit — primul servit și nu se blochează în fața secțiunii critice (cu condiția ca nici un procesor să nu înregistreze defectări și reveniri repetate, menținînd permanent *alege* $[i]$ pe unu). Creșterea nelimitată a valorilor $nr\ [i]$, alese de procese, face ca implementarea algoritmului să necesite adoptarea unor măsuri speciale de precauție.

2.1.2. REGIUNI CRITICE

Excluderea mutuală a secțiunilor critice ale unor procese diferite se bazează tot pe un mecanism de eliminare mutuală, aplicat însă la nivelul operațiilor elementare: inspectarea și modificarea valorii unei variabile comune. Mai mult, diferite alte forme de sincronizare între procese sînt posibile datorită excluderii mutuale. Avînd în vedere caracterul fundamental al acestui concept, precum și posibilitatea implementării sale la nivelul unor operații simple ori complicate, s-au propus o serie de construcții sintactice care să reprezinte — la nivelul limbajelor de programare utilizate — acțiuni mutual exclusive. Cele mai cunoscute sînt primitivele P și V [D-1], precum și regiunile critice [H-1.]. În continuare ne referim la a doua construcție, prezentînd caracteristicile sale principale, precum și modul de utilizare în rezolvarea unei alte probleme de sincronizare între procese, aceea a interferenței proceselor de citire și de scriere.

Notația

var v : **shared** T

declară o variabilă comună (shared) v de tip T. Procesele paralele pot inspecta și/sau modifica variabilele comune doar prin instrucțiuni numite regiuni critice. O regiune critică are forma:

region v **do** S,

care asociază instrucțiunea S cu variabila comună v. Regiunile critice care se referă la aceeași variabilă v se exclud reciproc în timp. Mai precis: (1) timpul în care un proces intră într-o regiune critică este finit; (2) la un moment dat, cel mult un proces, se poate afla într-o regiune critică; (3) un proces rămîne într-o regiune critică doar pentru o perioadă de timp limitată. De aici rezultă mai întîi că un proces poate intra într-o regiune critică imediat cînd nici un proces nu se află în regiuni critice. Dacă un proces se găsește într-o regiune critică, alte procese care încearcă să pătrundă în regiuni critice vor fi întîrziate; cînd procesul iese din regiunea critică, unul din procesele întîrziate va putea intra în regiunea sa critică. Regula după care se face selecția procesului ce urmează să intre în regiunea critică trebuie să nu întîrzie la nesfîrșit un proces, în favoarea proceselor mai urgente (regula poate fi, de exemplu, FIFO, dar nu LIFO); în afara acestei condiții nu se impune nici o altă restricție asupra ordinii specifice în care procesele intră în regiunile critice, ea depinzînd de implementare.

Folosind variabilele comune și regiunile critice, problema excluderii mutuale a operațiilor în timp capătă o rezolvare foarte simplă. Pentru exemplificare să considerăm că o resursă de calcul este utilizată în comun de n procese. Constrîngerea impusă este ca utilizarea resursei să se facă în regim de excludere mutuală. O soluție posibilă este ca operațiile fiecărui proces, referitoare la resursa comună, să fie încadrate într-o regiune critică. Descrierea acestei soluții este dată în algoritmul 2.4.

Algoritmul 2.4.

```
var R : shared boolean;
cobegin
   „P1“: repeat
            region R do * utilizează resursa;
            * rest proces P1
         forever
   .....
   „Pn“: repeat
            region R do * utilizează resursa;
            * rest proces Pn
         forever
coend
```

Corectitudinea soluției este evidentă, utilizarea în regim de excludere mutuală a resursei fiind asigurată prin mecanismul regiunilor critice.

Regiunile critice constituie un instrument simplu de sincronizare prin intermediul căruia execuția unui proces este întîrziată pînă cînd se îndeplinește o condiție particulară (condiția de excludere mutuală a operațiilor). Instrumentul de sincronizare complex, care permite întîrzierea procesului pînă la înfăptuirea unei condiții oarecare este cel al regiunilor critice condiționale. Acestea au la bază regiunile critice definite anterior și utilizează, în plus, o primitivă de sincronizare, **await**, care întîrzie execuția unui proces pînă cînd componentele unei variabile v comune satisfac o condiție B oarecare.

Primitiva **await** trebuie inclusă textual într-o regiune critică asociată cu variabila v:

```
var v : shared T
region v do
begin.....await B;.....end
```

Condiția B de sincronizare este o expresie booleană oarecare, putînd include elemente ale lui v.

Mecanismul de sincronizare caracteristic regiunilor critice se menține și în cadrul celor condiționale, astfel încît condiția de excludere mutuală este îndeplinită. Peste acest mecanism se suprapune efectul primitivei **await**, care poate fi executată de un proces doar în cadrul unei regiuni critice. Dacă atunci cînd procesul execută instrucțiunea **await** condiția B este îndeplinită, procesul continuă operațiile regiunii critice în care se află. Cînd însă condiția nu este îndeplinită, procesul trebuie întîrziat pînă la satisfacerea ei. În acest caz restricția de excludere mutuală se anulează, ca și cum procesul părăsește regiunea critică, astfel încît un alt proces poate intra într-o regiune critică asociată lui v. Deci secvența anterioară este posibil să se reproducă și, deci, la un moment dat pot exista mai multe procese întîrziate pînă la îndeplinirea unor condiții oarecare. Pe de altă parte, oricare proces care execută o regiune critică asociată lui v, este în măsură să modifice valorile componentelor acesteia, astfel încît una sau mai multe condiții să fie satisfăcute. Procesele întîrziate la aceste condiții devin pretendente la ocuparea regiunilor critice asociate lui v. Cînd un astfel de proces primește dreptul de intrare în regiunea critică, el continuă execuția cu instrucțiunea imediat următoare primitivei **await** care i-a provocat întîrzierea.

Fiecare proces poate modifica — prin execuția unei regiuni critice — valorile componentelor lui v. Ca urmare toate condițiile B ale proceselor întîrziate trebuie reevaluate ori de cîte ori se pune problema selecției unui proces pentru intrare în regiunea critică.

Descrierea modului de implementare a întîrzierilor proceselor la condițiile B și a excluderii acestora la execuția regiunilor critice se face în [H-1.].

O variantă posibilă pentru rezolvarea problemei anterioare de excludere mutuală prin utilizarea regiunilor critice condiționate este cea din algoritmul 2.5.

Algoritmul 2.5.

```
var R : shared boolean;
begin
   R : = true;
   cobegin
      P1 : repeat
                 * rest proces P1;
                 region R do
                    begin await R;
                       R : = false;
                    end
                 * utilizează resursa;
                 region R do R : = true;
           forever
      .....................
      Pn : repeat
                 * rest proces Pn;
                 region R do
                    begin await R;
                       R : = false;
                    end
                 * utilizează resursa;
                 region R do R : = true;
           forever
   coend
end
```

Se observă că în această soluție folosirea resursei comune în regim de excludere mutuală nu mai este realizată ca efect implicit al regiunilor critice (în urma includerii operațiilor referitoare la resursa comună în regiuni critice), ci prin utilizarea unei variabile R, — care reprezintă ocuparea resursei (R = = true-resursă liberă, R = false — resursă ocupată) — și încadrarea secțiunii ce conține operațiile de folosire a resursei în două regiuni critice, avînd ca efect „acapararea", respectiv „eliberarea" resursei, după tiparul:

acaparează resursă;
utilizează resursă;
eliberează resursă.

Ca efect, soluția de față nu aduce nimic nou față de cea anterioară cu regiuni critice „simple". Ea prezintă însă avantajul de a putea fi extinsă la tratarea unor probleme de sincronizare mai complicate.

Exemplul care urmează modelează relațiile dintre două grupuri de procese cu comportamente diferite față de o resursă comună. Procesele din primul grup, numite de citire, inspectează valorile componentelor unei baze de date comune; cele din al doilea grup, de scriere au dreptul în plus de primele, să modifice valorile componentelor. Din aceste motive mai multe procese de citire pot avea acces simultan la resursa comună (baza de date comună), în timp ce procesele de scriere utilizează resursa numai în regim de excludere mutuală.

Actualizarea cu promptitudine a bazei de date comune impune, de multe ori, ca procesele de scriere să aibă prioritate asupra celor de citire. Astfel, dacă un proces de scriere face o cerere de acces la resursa comună în timp ce ea este utilizată de procesele de citire, atunci nici un alt proces de citire (în afara celor care folosesc deja resursa) nu va mai căpăta dreptul de acces la resursă pînă ce solicitarea procesului de scriere nu este satisfăcută.

Soluționarea problemei sincronizării între cele două grupuri de procese este posibilă cu ajutorul tiparului menționat anterior. Se observă ușor că ocuparea resursei nu mai poate fi caracterizată printr-o singură variabilă logică R, ca mai înainte, fiind necesar să se distingă între ele mai multe situații. Enumerăm aceste situații, specificînd și modul lor de reprezentare în model, prin două variabile cu valori întregi, r și s, componente ale unei variabile v comune:

$r > 0$ — resursa este folosită de r procese de citire;
$r = -1$ — resursa este utilizată de un proces de scriere;
$r = 0$ — resursa nu este ocupată.

Simbolul s reprezintă numărul de procese de scriere care folosesc sau „doresc" să apeleze la resursă (din cele s procese, cel mult unul poate utiliza la un moment dat resursa). Ca urmare, resursa este posibil să fie alocată unui proces de citire dacă nici un proces de scriere nu dorește să o întrebuințeze ($s = 0$). Prin alocare, numărul de procese de citire ce utilizează resursa se mărește cu o unitate ($r := r + 1$). Deci operația *acaparează resursa* pentru un proces de citire este:

```
region v do
    begin await s = 0; r := r + 1
    end.
```

Pe de altă parte, eliberarea resursei de către un proces de citire are ca efect micșorarea cu 1 a numărului de procese de citire care utilizează resursa, adică:

```
region v do r := r − 1;
```

Cînd un proces de scriere dorește să folosească resursa, s trebuie crescut cu o unitate. Resursa poate fi alocată unui proces de scriere dacă este liberă ($r = 0$). Prin alocare resursa trece în utilizarea unui proces de scriere (r devine -1). Așadar, operația *acaparează resursa* pentru un proces de scriere este:

```
region v do begin s := s + 1;
                  await r = 0;
                  r := − 1
                  end.
```

În fine, eliberarea resursei de către un proces de scriere are ca urmare micșorarea cu o unitate a numărului de procese care întrebuințează sau doresc

să folosească resursa ($s: = s - 1$) și eliberarea efectivă a resursei ($r:=0$). Deci, operația *eliberează resursa* corespunzătoare unui proces de scriere este:

```
region v do begin s : = s−1;
                  r : = 0
            end.
```

Soluția problemei este descrisă în întregime în algoritmul 2.6.

Algoritmul 2.6.

```
var v : shared record r, s : integer end;
begin
    r:=0; s:=0;
    cobegin
        repeat „proces citire“
            * rest proces citire;
            region v do
                begin await s = 0
                      r := r + 1
                end
            * citește;
            region v do r := r−1;
        forever
        .....
        repeat „proces scriere“
            * rest proces scriere;
            region v do
                begin s:= s+1
                      await r=0;
                      r:= −1
                end
            * scrie;
            region v do
                begin s : = s−1;
                      r : = 0
                end
        forever
        .....
        coend
end.
```

2.1.3. MONITOARE

Așa cum s-a arătat, regiunile critice condiționale reprezintă un instrument flexibil de exprimare a diferitelor tipuri de interacțiuni între procese, dar suferă de o ineficiență crescută de implementare datorată necesității de reevaluare frecventă a condiției de sincronizare inclusă în primitiva **await.** Pentru a înlătura acest neajuns, Hoare [H-3.] și Brinch Hansen [H-1] au propus un alt concept, derivat din regiunile critice condiționale și din structurile abstracte de date, și anume cel de monitor.

Un monitor este o structură pasivă care încapsulează un grup de date împreună cu operațiile (procedurile) de acces la ele. Diferitele procese pot

testa și modifica datele unui monitor numai prin apelul procedurilor asociate. Deoarece monitorul este destinat utilizării într-un mediu distribuit, a fost necesară adoptarea unor măsuri de evitare a accesului simultan al mai multor procese la datele lui. Concret, procedurile unui monitor constituie secțiuni critice, doar una putînd fi executată la un moment dat. Interacțiunea proceselor diferite este realizată prin intermediul unor variabile condiție și a două operații de sincronizare, *wait* și *signal* (sau *delay* și *continue*, în notația lui Hansen). Operația *q.wait* blochează procesul executant, introduce identificatorul său în coada de așteptare asociată variabilei condiție *q* și eliberează totodată secțiunea critică. Operația *q. signal* activează un proces din coada de așteptare *q*, procesul care apelează operația este suspendat temporar pînă cînd procesul activat iese din monitor sau se blochează din nou; în acest moment execuția procesului apelant este reluată. Operația *q. signal* nu are nici un efect cînd coada *q* este goală.

Utilizarea monitoarelor va fi exemplificată cu ajutorul problemei proceselor de citire și de scriere care fac acces la o bază de date comune. Citirea din baza de date este încadrată între operațiile de inițiere a citirii și de terminare a citirii, iar scrierea în baza de date, între operațiile de inițiere a scrierii și de terminare a scrierii. Acestea reprezintă de altfel chiar procedurile asociate monitorului „baza-de-date", care mai cuprinde variabilele *r* și *s* (pentru care se păstrează semnificațiile din paragraful 2.1.2), precum și variabilele-condiție *r1* și *s1*, utilizate la sincronizarea proceselor de citire, respectiv de scriere.

Monitorul are descrierea dată în algoritmul 2.7.

Algoritmul 2.7.

```
monitor baza-de-date;
    var r, s: integer;
        r1, s1: condition;
    procedure initiere-citire;
        begin if s≠0 then r1. wait;
            r:=r+1; r1. signal
        end;
    procedure terminare-citire;
        begin r:=r−1;
            if r=0 then s1.signal
        end;
    procedure initiere-scriere;
        begin s:=s+1;
            if r≠0 then s1.wait;
            r:=−1
        end;
    procedure terminare-scriere;
        begin s:=s−1; r:=0;
            if s≠0 then s1. signal
                   else r1. signal
        end;
begin
    r:=0; s:=0
end;
```

Monitorul cuprinde o parte declarativă pentru date și proceduri și o parte de inițializare a datelor, care se execută la început, înainte de apelul oricărei proceduri. Este greu de apreciat cantitativ eficiența acestei soluții față de cea exprimată cu ajutorul regiunilor critice condiționate (pentru o tratare riguroasă a subiectului vezi lucrarea [C—1]). Totuși, putem observa că aici condițiile sînt testate explicit și, ca urmare, doar atunci cînd modificarea variabilelor poate duce la îndeplinirea lor. De exemplu, în procedura „terminare-citire" condiția $r=0$ este intim legată de operația anterioară de decrementare a lui r. De asemenea, ea nu declanșează automat evaluarea altor condiții relative la datele aceluiași monitor. Putem deci aprecia calitativ că soluția cu monitoare este mai eficientă decît cea cu regiuni critice. Totodată însă reclamă programarea explicită a unor acțiuni, care în cealaltă variantă erau executate automat. De exemplu, pentru activarea tuturor proceselor de citire ce așteaptă utilizarea bazei de date la un moment dat procesul de scriere activează (în procedura „terminare-scriere") doar unul dintre ele, urmînd ca fiecare proces de citire să activeze la rîndul său un altul (în procedura „inițiere-citire"), dacă există. Deși se apreciază că monitoarele conduc uneori la soluții a căror verificare este anevoioasă, ele s-au dovedit a fi un instrument util chiar în aplicații pentru rețele de calculatoare [H—2]. Un concept asemănător monitoarelor, dar orientat către o implementare mai eficientă, este cel de resursă compusă [C—1], la care se vor face referiri în paragraful 2.3.5.

2.1.4. RENDEZ-VOUS

Un concept cu totul deosebit de cele discutate anterior relativ la cooperarea proceselor prin comunicare este cel propus de Hoare [H—4] și cunoscut sub numele de rendez-vous. De data aceasta comunicarea între două procese secvențiale concurente este realizată direct, prin simple comenzi de intrare/ieșire. O comandă de intrare are forma generală: *sursă? variabilă*, unde *sursă* este identificatorul unui proces care furnizează o valoare, iar *variabilă* reprezintă variabila căreia urmează să i se atribuie valoarea în urma execuției comenzii. O comandă de ieșire are forma generală: *destinație*! *expresie*, în care *destinație* este identificatorul unui proces căruia i se transmite valoarea *expresiei*.

Comenzile de intrare/ieșire nu se execută niciodată singure, ci numai în perechi de comenzi corespondente. O pereche este formată dintr-o comandă de citire și una de scriere, ce aparțin unor procese diferite și îndeplinesc următoarele condiții: comanda de intrare din primul proces are ca sursă identificatorul celui de al doilea proces; comanda de ieșire din al doilea proces are ca destinație identificatorul primului proces; variabila din comanda de intrare corespunde (ca tip) expresiei din comanda de ieșire. Două comenzi corespondente se execută întotdeauna simultan, efectul fiind atribuirea valorii expresiei din comanda de ieșire, variabilei din comanda de intrare. Pentru a realiza execuția simultană procesul care întîlnește primul o comandă de intrare sau de ieșire se blochează pînă cînd celălalt proces întîlnește comanda corespondentă. Dacă al doilea proces își termină execuția fără a întîlni această comandă, execuția comenzii primului proces eșuează. Două sau mai multe procese se pot chiar bloca definitiv, în situația în care comenzile lor de intrare/ieșire nu corespund.

Interacţiunile mai complicate decît comunicarea directă între procese nu se mai realizează prin structuri pasive, ca în cazul monitoarelor, ci prin definirea unor procese cu funcţii speciale, care comunică cu mai multe procese utilizator. Pentru a ilustra acest concept printr-un exemplu se prezintă mai întîi, într-o formă simplificată, cîteva dintre notaţiile utilizate în limbajul definit de Hoare în lucrarea sa [H—4].

Un *proces* este reprezentat printr-un identificator urmat de o listă de comenzi, în forma *identificator*: *listă-de-comenzi*. El reprezintă o componentă a unei comenzi paralele, de forma [*proces* || *proces* || ... || *proces*], care specifică execuţia paralelă a proceselor componente. *Lista-de-comenzi* este o succesiune de comenzi simple sau structurate. O comandă simplă poate fi o atribuire, o comandă de intrare/ieşire sau o comandă nulă, în timp ce comanda structurată este posibil să fie o decizie, un ciclu sau (din nou) comandă paralelă. *Decizia* are forma unei succesiuni de *comenzi-cu-gardă* [D—2] : [*comandă-cu-gardă* □ *comandă-cu-gardă* ... □ *comandă-cu-gardă*], fiecare *comandă-cu-gardă* avînd la rîndul ei forma: *gardă* → *listă-de comenzi*.

O *gardă* conţine mai multe elemente, un element putînd fi o expresie logică sau o declaraţie (care introduce variabile avînd ca domeniu de valabilitate lista de comenzi asociată); garda se poate termina cu o comandă de intrare. Comanda cu gardă este executată numai dacă garda ei nu eşuează (cînd nici una din expresiile logice nu este falsă). Comanda de intrare de la sfîrşitul gărzii se execută numai conform regulilor obişnuite, în tandem cu o comandă de ieşire corespondentă. Execuţia deciziei constă în realizarea unei singure comenzi cu gardă (chiar dacă mai multe comenzi sînt valide). Cînd toate comenzile cu gardă eşuează, decizia eşuează la rîndul său.

Un *ciclu* are forma unei decizii precedate de un asterisc şi specifică un număr nelimitat de iteraţii ale comenzilor cu gardă constituente. Dacă toate gărzile eşuează, ciclul se termină fără efect. Altfel, decizia este executată şi întregul ciclu este reluat de la capăt.

În cele ce urmează se va descrie un proces care joacă rolul de tampon între un proces producător de mesaje şi unul consumator de mesaje, compensînd diferenţele dintre vitezele cu care mesajele sînt produse, respectiv consumate. Tamponul are o capacitate limitată, de 10 mesaje. Procesul consumator conţine perechea de comenzi *X*!*mai* ();*X*?*m*, reprezentînd transmiterea unui „semnal" cu constructorul „*mai*" şi nici o componentă, urmată de recepţia unui mesaj. Procesul producător conţine comanda *X*!*m* de transmitere a unui mesaj. Procesul-tampon *X* are descrierea dată în algoritmul 2.8.

Algoritmul 2.8

```
X ::
    tampon : (0..9) mesaj;
    in, out : integer ; in:=0; out:=0;
        *[in < out + 10; producător? tampon (in mod 10) → in:=in+1
            □ out<in; consumator? mai()→consumator! tampon (out mod 10);
                                        out:=out+1]
```

Funcţionarea procesului *X* este ciclică, la fiecare ciclu putînd fie accepta un mesaj de la producător (dacă tamponul conţine mai puţin de 10 mesaje), fie livra un mesaj consumatorului (în situaţia că tamponul nu este gol).

Procesul se termină cînd tamponul este gol (*in=out*) și procesul producător se termină. Procesul *X* joacă rolul unui monitor, fiind însă — ca oricare alt proces — o componentă activă a sistemului. Deși reprezintă o formă atractivă de sincronizare între procese, tehnica descrisă este potrivită pentru grupuri de procese rezidente într-un singur nod al sistemului teleinformatic, datorită constrîngerilor severe de timp impuse la execuția operațiilor de intrare/ieșire. Relaxarea acestor constrîngeri este realizată în tehnica la care se fac referiri în continuare, ce presupune cooperarea între procese paralele asincrone tot prin intermediul unor procese speciale, de gestiune a grupurilor de date comune. Orice proces care dorește să realizeze o anumită operație asupra datelor comune adresează procesului gestionar corespunzător un ordin, unde indică operația solicitată, urmînd să primească un răspuns la terminarea execuției operației. Un ordin care nu poate fi satisfăcut imediat de către gestionar este pus într-o coadă de așteptare, urmînd a fi luat în considerație într-un moment ulterior. Transmiterea și recepția ordinelor și răspunsurilor se fac prin comunicare de mesaje. Pentru a putea exploata eficient paralelismele posibile între acțiunile proceselor se înglobează în gestionar doar operațiile de sincronizare, rămînînd ca operațiile efective asupra datelor comune să fie realizate de alte procese, puse în execuție de gestionar (ca alternativă, operațiile pot fi executate chiar de procesele apelante, caz în care gestionarul joacă efectiv rolul unui monitor).

2.2. MODELE BAZATE PE TRANSMITERE DE MESAJE]

Într-un sistem teleinformatic cooperarea dintre procese situate în același sistem de calcul sau în sisteme de calcul diferite, se face — de regulă — prin schimb de mesaje. Deoarece sistemele de calcul se află la distanță unele de altele, întîrzierea de transmitere a mesajelor între procese nu este neglijabilă în raport cu timpul dintre două evenimente ale unui proces. Deci problemele de sincronizare între procese, deși similare celor prezentate în paragrafele anterioare, se rezolvă prin metode diferite, specifice sistemelor teleinformatice. Soluțiile cunoscute [L-3.] au la bază relația de ordine parțială definită prin „petrecut înainte", precum și extinderea ei la o ordonare totală a evenimentelor.

2.2.1. ORDONAREA EVENIMENTELOR ÎN SISTEMELE DISTRIBUITE

În vorbirea curentă se afirma că un eveniment *a* s-a produs înaintea altui eveniment *b* dacă momentul de timp la care s-a produs *a* precede momentul producerii lui *b*. Extinderea acestei accepții la sistemele teleinformatice întîmpină dificultăți, deoarece reclamă ca sistemele de calcul să conțină ceasuri de timp real, iar ceasurile să măsoare cu mare precizie timpul. O soluție mai realistă este aceea a introducerii unor ceasuri logice, care să marcheze ordinea evenimentelor fără a da cu exactitate momentele producerii lor. În acest scop se definește relația „petrecut înainte" pe mulțimea evenimentelor dintr-un sistem teleinformatic, notată cu →, ca fiind cea

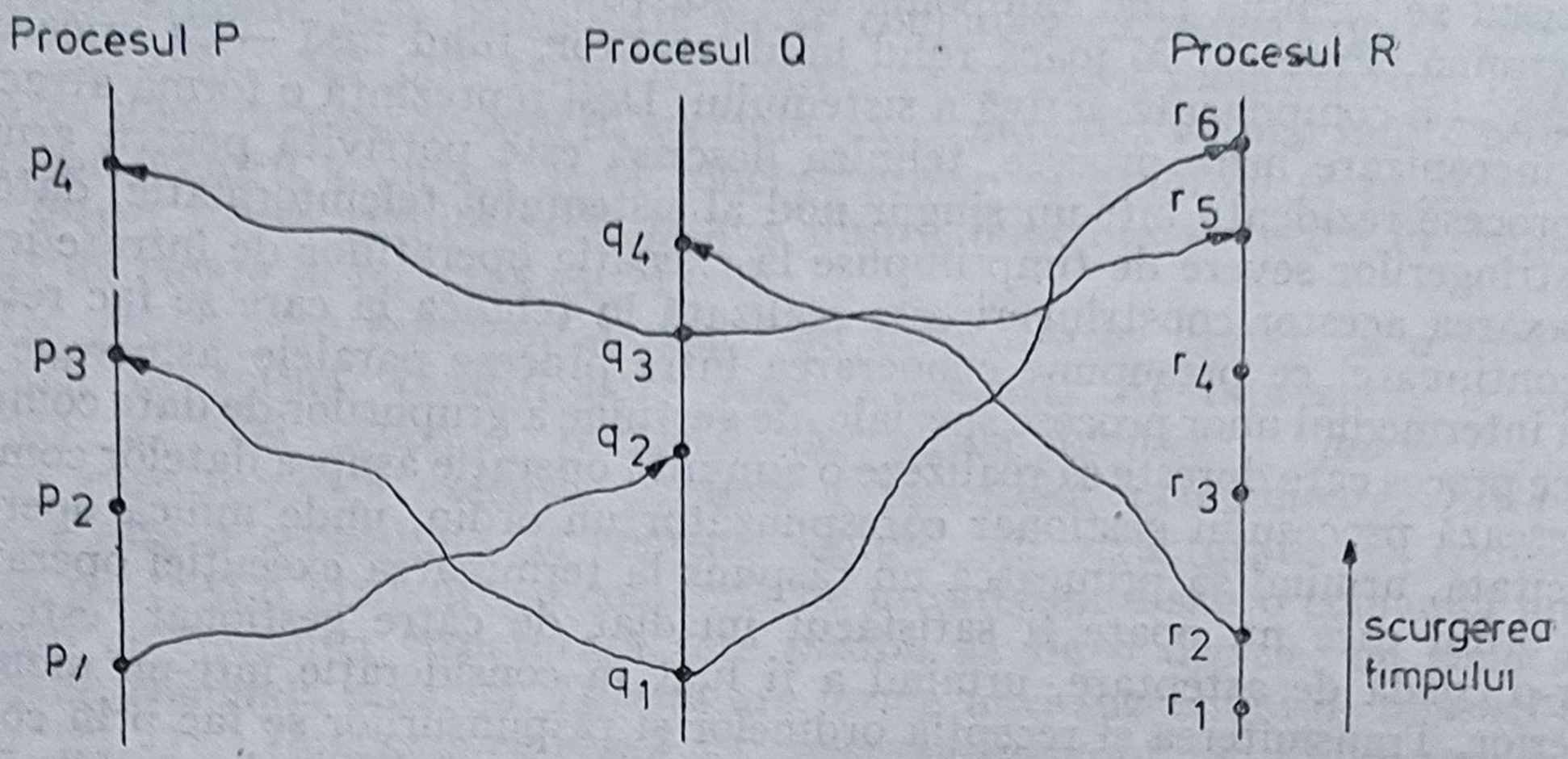


Fig. 2.5. Reprezentarea relației pentru trei procese — P, Q și R

mai slabă relație care satisface următoarele trei condiții: în situația că a și b sînt evenimente din același proces și a îl precede pe b, atunci $a \rightarrow b$; cînd a reprezintă transmiterea unui mesaj de către un proces, iar b, recepția aceluiași mesaj de către alt proces, înseamnă că $a \rightarrow b$; dacă $a \rightarrow b$ și $b \rightarrow c$, atunci $a \rightarrow c$. Două evenimente distincte, a și b, se numesc concurente cînd nici $a \rightarrow b$ și nici $b \rightarrow a$. În plus, $a \rightarrow a$ pentru orice eveniment a.

Relația $\rightarrow$ poate fi vizualizată în forma unei diagrame spațiu-timp (fig. 2.5). Spațiul este reprezentat pe orizontală, iar timpul pe verticală. Fiecare proces este redat printr-o linie verticală pe care sînt marcate evenimentele în ordinea producerii lor în timp. Liniile sinuoase denotă mesaje. După cum se observă, mesajele nu sînt recepționate neapărat în ordinea transmiterii lor. Comunicarea mai multor mesaje poate fi un singur eveniment, dar prin convenție recepția unui mesaj nu coincide cu transmiterea ori recepția nici unui alt mesaj. Tot prin convenție, dacă $a \rightarrow b$, atunci a și b sînt astfel reprezentate încît se poate ajunge de la a la b de-a lungul liniilor de procese și mesaje în sensul scurgerii timpului. Urmînd această regulă găsim că — de exemplu, în figura 2.5. — $p_1 \rightarrow r_6$.

Să considerăm în continuare o modalitate simplă de a introduce ceasuri logice într-un sistem de procese paralele. Pentru aceasta, în loc să se atașeze unui eveniment timpul real de producere, i se asociază un număr de ordine care să precizeze poziția sa în raport cu celelalte evenimente. Mai precis, pentru fiecare proces P_i se definește ceasul logic ca o funcție, C_i, care asociază oricărui eveniment a al lui P_i un număr de ordine, unic $C_i\ (a)$. Întregul sistem de ceasuri poate fi privit ca o funcție C, definită pe mulțimea tuturor evenimentelor cu valori în mulțimea numerelor naturale $\mathbf{N}$, ce asociază fiecărui eveniment a un număr $C\ (a)$, egal cu $C_i\ (a)$ dacă a este un eveniment al procesului P_i. Un astfel de sistem de ceasuri este corect cînd pentru orice evenimente a și b, dacă $a \rightarrow b$ atunci $C(a) < C(b)$. Ținînd seama de definiția relației $\rightarrow$, condiția de corectitudine a ceasurilor este echivalentă cu următoarele: (1) dacă a și b sînt evenimente ale aceluiași proces P_i și a se produce înaintea lui b atunci $C_i(a) < C_i(b)$; (2) cînd a reprezintă transmiterea unui mesaj de către procesul P_i, iar b recepția aceluiași mesaj de către procesul P_j, $C_i\ (a) < C_j\ (b)$.

Realizarea practică a unui sistem de ceasuri logice presupune folosirea pentru fiecare proces P_i a unei variabile V_i, care are valoarea $C_i(a)$ la producerea evenimentului a. Valoarea lui V_i se modifică între evenimente (schimbarea neconstituind ea însăși un eveniment), în așa fel încît să fie respectate condițiile (1) și (2) de corectitudine a sistemului de ceasuri. Prima condiție este îndeplinită, dacă fiecare proces P_i mărește V_i între oricare două evenimente succesive. Satisfacerea celei de a doua condiții reclamă o corelare între procesul emițător P_i și procesul receptor P_j ale aceluiași mesaj. O soluție este introducerea în fiecare mesaj m a unui cîmp adițional, T_m, care să ia valoarea C_i (a), reprezentînd momentul de transmitere a mesajului. La recepția mesajului P_j trebuie să actualizeze variabila V_j, astfel încît valoarea ei să depășească T_m.

2.2.2. PROTOCOL DE EXCLUDERE MUTUALĂ

În implementarea unui sistem distribuit este foarte utilă ordonarea totală a mulțimii evenimentelor sale [L—3]. De aceea, realizarea corectă a sistemului de ceasuri logice constituie o etapă importantă. Ea poate duce însă la situații în care două sau mai multe evenimente — aparținînd unor procese diferite — au același timp de producere. Pentru a putea decide în astfel de cazuri se recurge la o ordonare arbitrară a proceselor. Ca urmare, dacă evenimentele a, din procesul P_i și b din P_j au $C_i(a)=C_j(b)$, atunci dispoziția evenimentelor este dată de ordinea proceselor corespunzătoare. În continuare este descrisă utilizarea ordonării totale a mulțimii evenimentelor în rezolvarea problemei excluderii mutuale pentru sisteme teleinformatice. Rețeaua conține N noduri, fiecare putînd schimba mesaje cu toate celelalte prin intermediul unui serviciu de transport. În vederea realizării excluderii mutuale nodurile transmit între ele două tipuri de mesaje. Un nod care dorește să intre în secțiunea sa critică comunică mesaje de cerere — REQUEST — tuturor celorlalte noduri. Pentru a putea intra efectiv în secțiunea critică este necesar să primească de la fiecare nod cîte un mesaj de răspuns — REPLY. La recepția unui mesaj REQUEST un nod poate determina dacă el sau nodul de la care a primit cererea ar trebui să aibă acces mai întîi în secțiunea critică. Cînd el are prioritate mesajul REPLY este întîrziat; altfel este transmis imediat nodului ce a generat cererea. Fiecare nod execută același algoritm, prezentat în continuare (algoritmul 2.9) în forma unei structuri de date abstracte care conține o bază de date utilizată în comun de patru proceduri. Implementarea soluției presupune existența în fiecare nod a două procese. Unul conține secțiunea critică și execută procedura „invocă-excluderea-mutuală“ pentru a intra în ea și procedura „revocă-excluderea-mutuală“ la ieșire. Celălalt proces tratează mesajele recepționate și, în conformitate cu tipul acestora, execută procedura „recepție-REQUEST“ sau „recepție-REPLY“. Prin convenție, procedurile constituie regiuni critice (deși nu apare explicit în algoritm, fiecare procedură se consideră inclusă într-o regiune de forma: **region** bază-date-comună **do**....).

Algoritmul 2.9:

```
const  eu = numărul nodului curent;
       N = numărul de noduri din rețea;
var                          (* baza de date comună *)
  nr-secv,                   (* număr secvență propriu, inițial 0 *)
  nr-max,                    (* număr maxim secvență al nodului curent, inițial 0*)
  contor: integer;           (* numărul de mesaje REPLY așteptate *)
  cerere: boolean;           (* marcaj de cerere secțiune critică, inițial false; devine true cînd nodul cere acces la secțiunea critică *)
răspuns-întîrziat: array [1..N] of boolean;
                             (* elementul j este true cînd transmiterea mesajului REPLY către acest nod este întîrziată *)
procedure invocă-excludere-mutuală;
    begin
        cerere:=true;
        nr -secv :=nr-max+1;        (* alege un nr. secvență *)
        contor :=N-1;
        for j:=1 to N do
            if j<>eu then (* transmite mesaj REQUEST tuturor celorlalte noduri *)
        send (REQUEST (nr-secv, eu), j);
        await (contor=0)                    (* așteaptă mesaj REPLY *)
    end;                                    (* de la fiecare nod *)
procedure revocă-excludere-mutuală;
    begin
        cerere:=false;
        for j:= 1 to N do
        if răspuns-întîrziat [j] then (* transmite mesaj REPLY *)
                                      (* nodului j *)
            begin răspuns-întîrziat [j] :=false;
                  send (REPLY, j)
            end
    end;
procedure recepție — REQUEST (k, j : integer);
                             (* k — număr secvența din cerere
                                j — număr nod solicitant *)
    var întîrzie: boolean;
    begin
        nr-max:=maxim (nr-max, k);
        întîrzie:= cerere and ((k> nr-secv) or
                   (k=nr-secv and j > eu));
        if întîrzie then      (* dacă nodul curent are prioritate *)
                   răspuns-întîrziat [j]:=true
        else
                   send (REPLY, j)
    end;
procedure recepție-REPLY;
    begin
        contor:=contor—1
    end.
```

Algoritmul asigură excluderea mutuală, nodurile A și B — oricare ar fi ele — nu se pot afla simultan în secțiunile lor critice. Pentru a demonstra acest lucru se presupune contrariul, și anume, că ambele procese sînt în secțiunile lor critice. Deci amîndouă au primit cîte un mesaj REPLY de la fiecare din celelalte noduri. Sînt posibile mai multe situații.

Nodul B transmite mesajul REPLY nodului A înainte de alegerea propriului său număr de secvență. Ca urmare, B alege un număr de secvență mai mare decît cel al lui A. Cînd recepționează mesajul REQUEST transmis de B nodul A va întîrzia transmiterea mesajului REPLY, deoarece are „*cerere*=false“ și un număr de secvență mai mic. Așadar nu este posibil ca B să fi primit mesaje REPLY de la nodul A.

Nodul A transmite mesajul REPLY nodului B înainte de alegerea propriului său număr de secvență. Situația este simetrică celei din primul caz.

Ambele noduri transmit REPLY unul către celălalt după alegerea propriilor numere de secvență. Ca urmare, fiecare are „*cerere*=true“ cînd primește cererea celuilalt. Nodurile vor compara numerele de secvență și pe cele de ordine, obținînd rezultate diferite. Deci, unul din noduri va întîrzia transmiterea mesajului REPLY către celălalt.

Se poate arăta, de asemenea, că procesele nu se blochează definitiv înaintea secțiunilor critice. Presupunînd că mai multe noduri transmit mesaje REQUEST, atunci ele vor primi mesaje REPLY de la nodurile fără cereri și, în plus, datorită ordonării totale a cererilor (după perechea de valori: număr de secvență, număr de nod), unul din ele va recepționa mesaje REPLY de la toate celelalte noduri care au cereri. Acest nod va intra în secțiunea sa critică, deci nodurile nu se pot bloca definitiv înaintea secțiunilor lor critice. Rezultă totodată că datorită ordonării totale a cererilor un nod nu poate fi oprit la nesfîrșit să intre în secțiunea critică, în timp ce alte noduri intră și ies în/din propriile lor secțiuni critice. Și în algoritmul de față numerele de secvență cresc, teoretic, nelimitat. Pentru a evita neajunsurile la implementarea lui se recurge la limitarea domeniului numerelor de secvență prin tehnici comune protocoalelor nivelurilor inferioare. Deoarece numerele de secvență cresc cu cel mult o unitate la fiecare mesaj REQUEST, iar o cerere nu poate fi admisă cît timp există o altă cerere cu număr de secvență mai mic, rezultă că numerele de secvență aparțin domeniului de la x la $x+N-1$. Deci este posibil ca numerele de secvență să fie memorate modulo M, unde $M \geqslant 2N-1$. Cînd se face o comparație, numărul mai mic trebuie crescut cu M dacă diferența este de cel puțin N.

Algoritmul poate fi ușor modificat pentru a rezolva problema proceselor de scriere și de citire, cele de scriere fiind prioritare. Modificarea constă în aceea ca procesele de citire să nu întîrzie niciodată cereri ale altor procese de citire, ci să transmită imediat mesaje REPLY. Procesele de scriere urmează algoritmul nemodificat.

În fine, menționăm că algoritmul reclamă transmiterea a $2(N-1)$ mesaje la fiecare invocare a unei secțiuni critice și anume $N-1$ mesaje REQUEST și $N-1$ mesaje REPLY. Sînt cunoscuți și algoritmi care reclamă un număr mai mic de mesaje pentru fiecare invocare a unei secțiuni critice.

2.2.3. PROTOCOL PENTRU UN SISTEM DISTRIBUIT DE REZOLVARE DE PROBLEME

Algoritmul prezentat anterior constituie un bun exemplu al modului în care un protocol (în acest caz un protocol de excludere mutuală) este implementat prin schimb de mesaje între procesele sistemului teleinformatic. Aceleași principii stau la baza realizării și a altor protocoale de nivel înalt, utilizate în aplicații diverse, cum ar fi poștă electronică, transfer electronic de fonduri, sisteme de rezervare de locuri, sisteme cu introducere de lucrări de la distanță etc. În continuare prezentăm caracteristicile esențiale ale unui alt protocol, de cooperare între procesele unui sistem distribuit de rezolvare de probleme. Așa cum se va vedea, de data aceasta ordonarea evenimentelor nu mai joacă rolul important pe care îl are în rezolvarea problemelor din categoria excluderii mutuale.

Sistemul constă dintr-un număr de surse de cunoștințe (proceduri, mulțimi de reguli de producție etc.) localizate în nodurile unui sistem teleinformatic. Nici una dintre sursele de cunoștințe nu posedă suficiente informații pentru a putea rezolva în întregime o problemă. Ca urmare, sursele de cunoștințe cooperează între ele, într-o manieră descentralizată (nu există un control centralizat și nici date globale centralizate), prin schimb de mesaje, în conformitate cu un protocol de nivel înalt. Un astfel de sistem cînd primește o problemă de rezolvat trebuie să găsească modalitatea convenabilă de împărțire a sarcinii între nodurile sale. Cu alte cuvinte este necesar să fie selectate cele mai „competente“ noduri ale sistemului teleinformatic, iar încărcarea lor, cît mai uniformă. Una dintre căile de realizare a acestor deziderate este cea a contractelor. În acest scop, nodurile rețelei se împart în două categorii: *administrator* și *contractant*. Administratorul răspunde de coordonarea execuției sarcinii, în timp ce contractantul răspunde de execuția efectivă a sarcinii. Un nod poate juca simultan ambele roluri, pentru contracte diferite. Stabilirea contractelor presupune o înțelegere mutuală între administratori și contractanți. Contractanții disponibili evaluează anunțurile de sarcini făcute de diferiți administratori și prezintă oferte pentru acele sarcini pe care ar putea să le execute. Administratorii evaluează ofertele și „încheie contracte“ cu nodurile contractante cele mai potrivite. Procesul de negociere poate continua: oricare dintre nodurile contractante este în măsură să împartă sarcina în subsarcini și să încheie contracte pentru ele, în calitate de administrator, cu alte noduri. Aceasta duce la o structură ierarhică de control, tipică sistemelor cu distribuire a sarcinilor. Pentru a înțelege mecanismul realizării contractelor va fi descris procesul de negociere pe un exemplu. El se referă la un sistem distribuit de supraveghere, care constă dintr-o rețea de noduri de sesizare și noduri de prelucrare, răspîndite pe o zonă geografică întinsă. Sarcina sistemului este aceea de a construi și de a actualiza permanent o hartă a traficului de autovehicule în zona respectivă, operații ce necesită interpretarea și prelucrarea unei mari cantități de informație culeasă de senzori. Prin utilizarea protocolului de contractare se creează condițiile necesare pentru configurarea dinamică a sistemului de senzori, avînd în vedere factori cum sînt: numărul de noduri de sesizare și de prelucrare disponibile, localizările lor, ușurința cu care ele pot comunica etc.

Sarcina în favoarea căreia se face negocierea, numită *semnal*, constă în culegerea datelor furnizate de senzori și extragerea caracteristicilor semnalelor.

Către : *
Dela : 25
Tip : ANUNȚ SARCINĂ
Contract : 22-8-87
Rezumat sarcină :
TIP SARCINA SEMNAL
POZIȚIE LAT 47N LONG 17E
Specificație eligibilitate :
TREBUIE SĂ POSEDE SENZOR
TREBUIE SĂ POSEDE POZIȚIE ZONA A
Specificație ofertă :
POZIȚIE LAT LONG
FIECARE SENZOR NUME TIP
Timp expirare :
28 1730z FEB 1987

obs. * semnifică toate nodurile

Fig. 2.6. Anunțul de sarcină.

Administratorii acestei sarcini sînt nodurile de prelucrare, fără facilități de sesizare, dar în măsură să preia caracteristicile semnalelor de la nodurile de sesizare, în vederea prelucrării. Din punctul lor de vedere este de dorit ca mulțimea de contractanți să aibă dispunerea geografică și tipurile de senzori anumite, iar din cel al contractanților, administratorii să fie situați la distanță mică, pentru a realiza o comunicare cît mai economică. Împăcarea acestor puncte de vedere diferite constituie unul din obiectivele procesului de negociere.

Negocierea începe prin transmiterea de către fiecare dintre administratori a unui mesaj de *anunț de sarcină*, de genul celui prezentat în figura 2.6. Fiecare mesaj al protocolului are o serie de cîmpuri cu informații specifice. Informația care umple diferitele cîmpuri este codificată într-un limbaj simplu, comun tuturor nodurilor. Anunțul de sarcină din exemplul de față este adresat tuturor nodurilor din rețea. Se poate utiliza în egală măsură o adresare focalizată a unei submulțimi a nodurilor sau a unui singur nod.

Anunțul de sarcină are patru cîmpuri principale. Cîmpul *rezumat sarcină* este o scurtă descriere a sarcinii de executat. Ea permite unui nod să clasifice sarcina în raport cu alte sarcini enunțate. Cîmpul *specificație eligibilitate* este o listă de criterii pe care trebuie să le satisfacă un nod pentru a putea prezenta o ofertă. Rolul său este de a împiedica nodurile să facă oferte ce sînt în mod evident neacceptabile. Cîmpul *specificație ofertă* constituie un tipar conform căruia nodurile trebuie să-și prezinte ofertele. Prin intermediul său un nod administrator precizează informațiile ce le consideră relevante și în raport cu care va categorisi ofertele. În fine, cîmpul *timp expirare* reprezintă o limită pentru recepționarea ofertelor. El presupune existența unui mecanism de sincronizare globală între noduri. În orice caz, în procesul de negociere timpul nu este o mărime critică. De exemplu, sosirea unor oferte după timpul de expirare nu provoacă funcționări anormale ale sistemului, ci doar alegerea unei mulțimi suboptimale de contractanți.

În exemplul dat anunțul de sarcină specifică tipul sarcinii și poziția nodului de prelucrare. El indică totodată că nodul de sesizare eligibil trebuie să aibă senzori și să fie situat în zona *A*. Se precizează că oferta este necesar să precizeze poziția nodului ofertant, precum și numele și tipul fiecărui senzor.

Către : 25
Dela : 42
Tip : OFERTA
Contract : 22-8-87
Rezumat nod :
POZIȚIE LAT 60N LONG 4V
SENZOR NUME S1 TIP S
SENZOR NUME S2 TIP S
SENZOR NUME T1 TIP T

Fig. 2.7. Oferta.

La recepția unui anunț de sarcină un nod verifică specificația de eligibilitate și, dacă poate face o ofertă, clasifică anunțul în raport cu alte anunțuri similare (pentru fiecare tip de sarcină, un nod păstrează o listă de anunțuri recepționate). Recepția și clasificarea anunțurilor se fac în paralel cu execuția sarcinii curente. La terminarea acesteia, nodul devine disponibil pentru o altă sarcină. El poate alege (după un procedeu oarecare) unul din anunțurile recepționate și neexpirate, pentru a face o ofertă. În aceeași măsură el are posibilitatea să decidă așteptarea altor anunțuri. În primul caz nodul transmite o ofertă de genul celei din figura 2.7.

Cîmpul *rezumat nod* din ofertă conține informațiile cerute de nodul administrator, în forma dată în cîmpul *specificație ofertă* din anunțul de sarcină. El mai poate include o serie de cerințe (de forma: CERE PROCEDURA NUME TEST) prin care se indică nodului administrator că, în vederea execuției sarcinii, este nevoie de informații suplimentare. Aceste informații trebuie să fie transferabile și să nu fie incluse în specificația de eligibilitate din anunțul de sarcină (într-un enunț de forma TREBUIE SĂ POSEDE...).

În exemplul dat oferta specifică poziția nodului ofertant, împreună cu numele și tipurile senzorilor săi.

Administratorul clasifică ofertele recepționate. Dacă o anumită ofertă este considerată satisfăcătoare, contractul poate fi încheiat imediat prin transmiterea către ofertantul corespunzător a unui mesaj de genul celui din figura 2.8. Cîmpul *specificație sarcină* conține datele necesare începerii execuției sarcinii, împreună cu informațiile adiționale solicitate de contractant. Dacă timpul expiră și nu s-a încheiat încă contractul, administratorul poate transmite mesaje de acord nodurilor cu oferte acceptabile, încheind cu ele contracte, sau un nou anunț de sarcină. Pe parcursul execuției sarcinii sînt situații cînd nodul contractant și cel administrator schimbă între ele mesaje de diferite tipuri. Acestea pot fi mesaje de informație utilizate ca mijloc general de comunicare, rapoarte sau mesaje de terminare. Rapoartele sînt fie parțiale fie finale. Ele conțin rezultatele parțiale ale execuției ori rezultatele finale. Mesajele de terminare sînt folosite de administrator pentru a determina încetarea execuției contractului indicat și a subcontractelor corelate cu acesta.

Către : 42
Dela : 25
Tip : ACORD
Contract : 22-8-87
Specificație sarcină :
SENZOR NUME S1
SENZOR NUME S2

Fig. 2.8. Acordul.

Ca orice protocol de nivel înalt, protocolul de contractare a fost gîndit prin prisma atingerii unor obiective privind viteza de lucru a sistemelor distribuite, siguranța în funcționare, extensibilitatea lor și altele.

În vederea realizării unei viteze corespunzătoare s-a urmărit evitarea gîtuirilor care ar putea apare prin concentrarea sarcinilor de prelucrare în anumite noduri sau prin saturarea unor canale de comunicație. În acest sens s-a recurs la o distribuire a surselor de cunoștințe, a datelor și a controlului, iar prin procesul de negociere s-au creat premisele pentru o distribuire a sarcinilor. Totodată protocolul include elemente menite să reducă traficul pe canalele de comunicație. La aceasta contribuie în primul rînd alegerea cîmpurilor în diferite tipuri de mesaje, cum ar fi: *specificație eligibilitate* din anunțul de sarcină ori *rezumat nod* din ofertă. În al doilea rînd, interacțiunile focalizate contribuie la reducerea traficului în cazurile în care nu este necesară aplicarea mecanismului complex de negociere.

Creșterea siguranței în funcționare este determinată de distribuția controlului, procesul de negociere neputînd fi blocat de defectarea unor noduri ale rețelei. De exemplu, în cazul defectării unui nod contractant administratorul poate reanunța sarcina, încredințînd-o altui nod.

Procesul de negociere permite, de asemenea, extinderea unei rețele în vederea includerii de noduri noi. Acestea nu trebuie să-și anunțe explicit prezența, putînd solicita oricînd sarcini apărute ulterior. Protocolul este potrivit pentru aplicații unde definirea unei ierarhii de sarcini reprezintă o cale naturală de rezolvare (cum ar fi căutarea euristică) și în care efortul necesar pentru răspîndirea sarcinii este mai mic decît cel de execuție efectivă a sarcinii.

2.2.4. PROTOCOL BAZAT PE TRANSMITEREA UNUI PRIVILEGIU

Protocolul de excludere mutuală descris în continuare se bazează pe faptul că fiecare proces poate fi unic identificat în raport cu celelalte procese, ceea ce permite stabilirea unei ordini între ele. În acest scop este posibil ca procesele să fie organizate într-un inel, fiecare avînd un predecesor unic și un succesor unic. De-a lungul inelului circulă în permanență un mesaj de control numit *jeton* sau *token*. Procesul care se află în posesia jetonului are dreptul de a executa secțiunea sa critică, după care jetonul este transmis procesului succesor. Evident, dacă procesul curent nu intenționează să execute secțiunea critică jetonul este transmis direct succesorului. Pentru ca acest protocol să fie utilizat în circumstanțe reale este necesar în primul rînd ca inelul să fie reconfigurabil, astfel încît funcționarea lui să nu fie afectată de căderea unui proces. În egală măsură însă trebuie evitată pierderea sau duplicarea jetonului. Prezentăm în continuare un protocol de detecție a pierderii jetonului și de regenerare a lui. Conform acestui protocol, pe inel se transmit, în afara jetonului de control, jetoane de alegere, fiecare conținînd identificatorul (numele) procesului care-l generează. Jetoanele circulă pe inel după politica FIFO. De asemenea, fiecare proces are un ceas ce produce un semnal de time-out după scurgerea unei cuante de timp predeterminate. Ceasul este reinițializat la fiecare recepție a unui jeton de orice fel.

Funcționarea procesului este caracterizată de două stări: normală și de alegere, premergătoare generării unui nou jeton de comandă. Procesul poate trece dintr-o stare în alta la producerea unuia din următoarele evenimente: 0 — scurgerea cuantei de timp (semnal de time-out); 1 — recepția unui jeton de control; 2 — recepția unui jeton de alegere cu un identificator mai mic decît acela al procesului curent; 3 — recepția unui jeton de alegere cu un identificator mai mare decît acela al procesului curent; 4 — recepția jetonului de alegere generat de procesul curent (după parcurgerea completă a inelului).

Protocolul prevede următoarele acțiuni ale proceselor, ca reacție la producerea diferitelor evenimente: la scurgerea cuantei de timp, procesul generează un jeton de alegere cu identificatorul său și intră în faza de alegere; la recepția jetonului de control revine în starea normală; la recepția unui jeton de alegere, înregistrează numele din jeton într-o listă de procese L, locală procesului curent; cînd acesta este propriul său jeton, îl extrage din inel și, dacă identificatorul procesului curent este egal cu min L (este cel mai mic din lista de procese L), atunci generează un nou jeton de control. Comportarea procesului poate fi urmărită în tabelul 2.1, care reprezintă tabela sa de tranziții. Pentru fiecare stare și eveniment posibile se indică starea următoare, precum și acțiunea executată de proces. De remarcat că înregistrarea proceselor în lista L este suplinită prin următorul mecanism simplu: la recepția unui jeton de alegere cu identificator mai mare procesul curent rămîne în starea b; la recepția unui jeton de alegere cu identificator mai mic, procesul curent trece în starea a, un alt proces urmînd să genereze jetonul de control.

TABELUL 2.1

stare \ eveniment	0	1	2	3	4
a	b, A	a	a	a	a
b	b	a	a	b	a, B

Pentru a arăta că jetonul de control nu poate fi generat de două procese diferite se introduc următoarele notații: $T(JC,x)$ — momentul recepției jetonului de control de către procesul x; $T(JA(y),x)$ — momentul recepției jetonului de alegere al procesului y de către procesul x; $T(x,0)$ — momentul generării jetonului de alegere de procesul x; $T(x,x)$ — momentul producerii evenimentului 4; $\neg k$ — absența evenimentului k. Demonstrația se face prin reducere la absurd. Să presupunem că două procese x și y, avînd identificatorii în relația $x < y$, generează în același timp cîte un nou jeton de control. Procesul y generează un jeton de control numai dacă starea sa la momentul $T(y,y)$ este b. Aceasta implică $\neg 1$ și $\neg 2$ între $T(y,0)$ și $T(y,y)$. După un raționament similar și deoarece $x<y$, pentru procesul x avem $\neg 1$ între $T(x,0)$ și $T(x,x)$. Dar $\neg 2$ între $T(y,0)$ și $T(y,y)$, înseamnă $T(JA(x),y) > T(y,y)$ pentru procesul y sau — cu alte cuvinte — procesul y generează jetonul de control înainte de a recepționa jetonul de alegere al procesului x. Ordinea jetoanelor pe inel fiind FIFO, rezultă că jetonul de control precede

jetonul de alegere al lui x. Pe de altă parte $\neg 1$ între $T(x, 0)$ și $T(x, x)$ înseamnă $T(JC,x) > T(x,x)$, unde jetonul de control recepționat de x este cel generat de y. Rezultă că se recepționează jetonul de alegere propriu înainte de jetonul de control, ceea ce este absurd. Deci jetonul de control nu poate fi generat de două procese diferite. Noul jeton de control este generat într-un interval finit de timp, a cărui limită superioară poate fi calculată pe baza întîrzierilor de transmitere a jetoanelor între procese. Protocolul este robust, eventuale erori înregistrate pe parcursul fazei de alegere putînd cel mult să mărească intervalul de generare a unui nou jeton de control.

2.3. CARACTERISTICI ALE SISTEMULUI DE OPERARE RELATIVE LA GESTIUNEA PROCESELOR

O situație des întîlnită în sistemele teleinformatice este cea în care mai multe mecanisme de sincronizare coexistă simultan, nucleul sistemului de operare distribuit furnizînd serviciile necesare implementării acestora. Din fericire toate mecanismele discutate pot fi realizate pe baza aceluiași set de primitive (se serviciu), a căror descriere este făcută în prima parte a secțiunii curente. În final este exemplificată utilizarea lor într-o construcție similară monitoarelor, avînd ca scop realizarea unor tipare diverse de cooperare între procese.

În algoritmii descriși anterior s-a considerat că diferitele procese evoluează în timp asincron unele față de altele, ca și cum fiecare ar avea propriul său procesor pe care să fie executat. În realitate numărul procesoarelor existente (același de multe ori cu numărul de sisteme de calcul) este mult mai mic decît cel al proceselor cooperante. Ca urmare, pentru a realiza efectul menționat în sistemele de operare ale calculatoarelor sînt utilizate unele mecanisme.

Resursele unui sistem de calcul pot fi împărțite între mai multe procese fie pentru a executa procesul pînă la terminare, fie întrețesut cu alte procese. Ambele variante presupun că în anumite momente procesorul trebuie să decidă dacă un proces va fi executat în continuare sau urmează să fie înlocuit cu un alt proces. Aceasta se face în conformitate cu un algoritm de planificare. În scopul simplificării, planificarea se face pentru fiecare resursă sau tip de resursă în parte. Aceasta conduce la structurarea funcțiilor sistemului de operare în mai multe niveluri, la fel ca în modelul arhitectural al sistemelor teleinformatice. În varianta cea mai răspîndită, împărțirea procesorului se efectuează în cadrul nucleului sistemului de operare. Deasupra acestui nivel lucrurile apar ca și cum fiecare proces ar avea propriul său procesor (virtual) pe care este executat. Procesul sau procesorul ce intervine într-un sistem teleinformatic poate fi specificat prin descriptori care conțin elementele semnificative referitoare la identitatea și starea acestora.

2.3.1. REPREZENTAREA PROCESORULUI

Un procesor (real) execută la un moment dat acțiuni ale unui singur proces. Dacă acțiunile mai multor procese trebuie executate pe același procesor, acesta este alocat succesiv proceselor solicitante. Spunem despre pro-

cesul ale cărui acțiuni sînt executate de procesor că este în *execuție*, iar despre cele ce așteaptă execuția că sînt *pregătite*. Procesele pregătite sînt enumerate într-o *listă de procese în așteptare* (*lpa*), alegerea unui proces din această listă pentru a fi trecut în execuție fiind realizată conform unui algoritm *planificator*. Informațiile necesare procesorului pentru controlul execuției procesului selectat formează vectorul de stare.

Convenim să reprezentăm descriptorul unui procesor printr-o înregistrare cu următoarele componente:

- *planificator* — procedură de planificare; valoarea componentei este indicatorul unei rutine de planificare a proceselor din *lpa*;
- *lpa* — referință la lista proceselor în așteptare; fiecare element al listei este o referință la un proces;
- *procex* — referință la procesul în execuție;
- *vectorstare* — ansamblul informațiilor necesare procesorului pentru controlul execuției unui proces.

Utilizăm notația

ucp: procesor

pentru a declara că *ucp* este descriptorul unui procesor. Oricare dintre componentele descriptorului unui procesor poate fi referită printr-o variabilă selectată, în conformitate cu regulile proprii tipului înregistrare. În particular dacă *ucp* este descriptorul unui anumit procesor, apelul planificatorului său va fi specificat în notația *ucp. planificator*.

Funcția planificatorului unui procesor este de a pune în execuție procese din *lpa*. Includerea lui în descriptorul procesorului permite evidențierea algoritmului conform căruia se realizează această funcție, înțelegerea și proiectarea lui ușoară și eventual înlocuirea cu alte variante pe parcursul proiectării sau realizării sistemului. Variantele de planificare a procesorului rezultă din combinarea unor proceduri diverse de planificare cu diferiți algoritmi de acces la elementele din *lpa*. Descrierea planificatorului va fi prezentată ca o procedură, implementarea acestuia într-un sistem de calcul putînd fi un program, un echipament sau o combinație a acestora.

Multiplexarea procesorului la execuția diferitelor procese necesită existența unor informații pe baza cărora să se poată realiza trecerea de la un proces la altul *(vector-stare)*. Presupunem că procesorul real este dotat cu suportul necesar memorării acestor informații (registre interne) și cu mecanismele de salvare și regăsire a lor. Convenim ca operațiile de salvare și regăsire să fie reprezentate de două proceduri standard, ale căror apeluri sînt:

memoreazăstrare *(proc)*

și

încarcăstare *(proc)*,

unde *proc* identifică procesul a cărui stare este salvată sau reîncărcată.

2.3.2. REPREZENTAREA PROCESULUI

În paragraful 2.3.1 s-a arătat că un proces ale cărui acțiuni sînt executate de un procesor este în starea de execuție. Există mai multe situații în care

acțiunile unui proces nu sînt, sau nu pot fi executate de nici un procesor al sistemului. Aceste situații sînt determinate de:

- existența unui număr redus de procesoare (mai mic decît numărul de procese din sistem);
- restricțiile de acces impuse de utilizarea în comun a unor resurse (limitarea numărului de procese care execută simultan acțiuni asupra resurselor comune);
- interacțiunile explicite între procese, un proces putînd suspenda execuția altui proces.

Procesul în așteptarea execuției pe un procesor este *pregătit*. Acțiunile proceselor pregătite ar fi executate, în ipoteza existenței unui număr de procesoare egal cu numărul de procese din sistem. Procesele pregătite sînt organizate în lista *lpa* asociată procesorului pe care urmează să fie puse în execuție.

Procesul aflat în așteptarea producerii unui eveniment este blocat. În particular, un proces se poate bloca în așteptarea recepționării unor mesaje transmise de alte procese. Considerăm că un proces se blochează numai prin acțiunile sale proprii. El este scos din starea de blocare de un alt proces, ale cărui acțiuni determină producerea evenimentului așteptat. Acest proces poate fi cel care transmite mesajul așteptat de prccesul blocat.

Procesul poate fi *suspendat* prin acțiunea explicită a unui alt proces. Dacă acțiunea de suspendare se execută atunci cînd procesul la care se referă este blocat, starea acestuia devine *blocatsuspendat*. Procesul rămîne înscris în lista în care se găsea în momentul suspendării. Restricția de suspendare poate fi ridicată printr-o acțiune explicită de „trezire" executată de un alt proces.

Stările și operațiile ce modifică starea unui proces sînt rezumate în figura 2.9.

Descriptorul procesului trebuie să reflecte starea acestuia, precum și circumstanțele care determină această stare. Convenim ca el să fie reprezentat printr-o înregistrare cu următoarele componente:

- *stare* — conform celor prezentate anterior, poate avea una din valorile 'execuție', 'pregătit', 'blocat', 'blocatsuspendat', 'suspendat';
- *procesor* — referință la procesorul care execută acțiunile procesului;

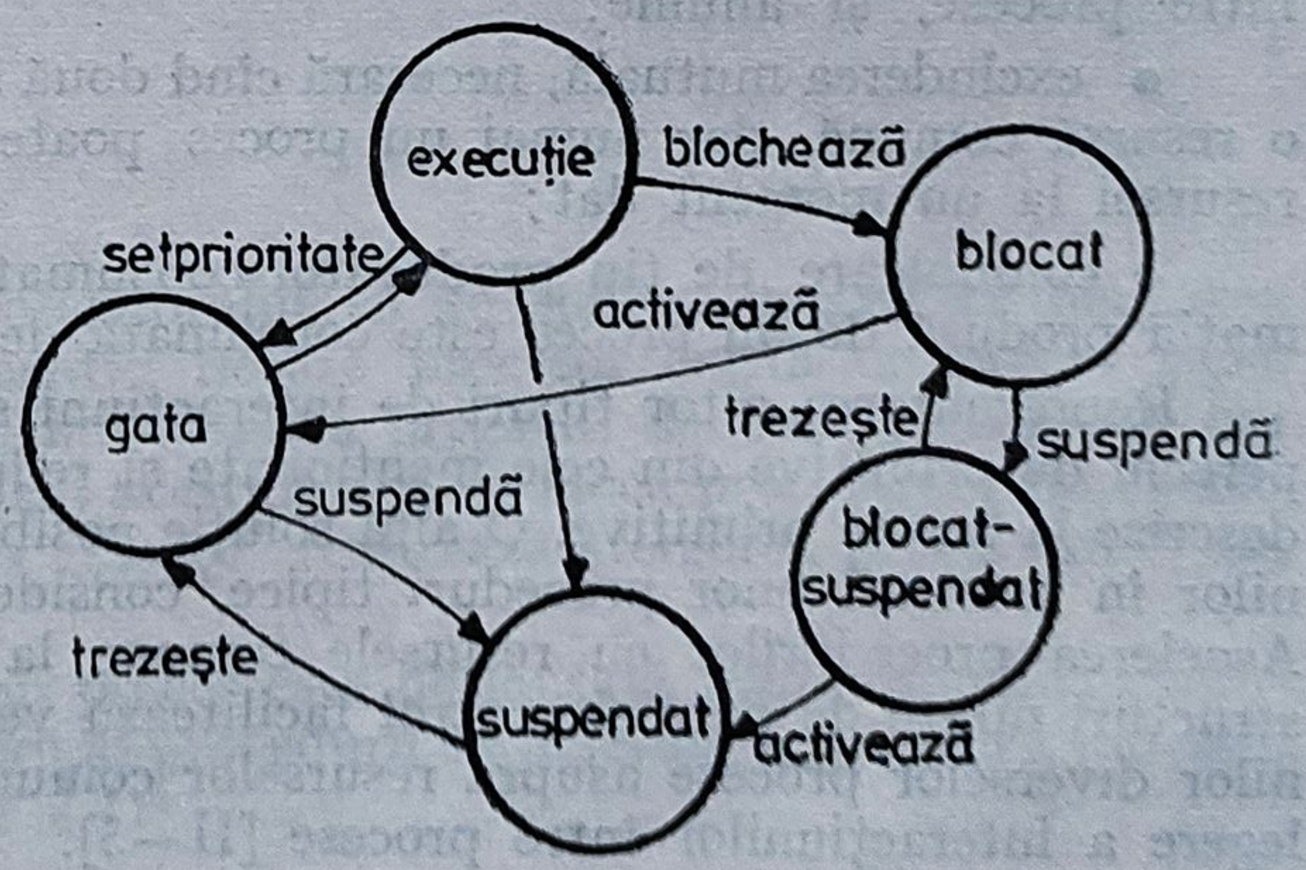


Fig. 2.9. Stările proceselor și operațiile asupra lor.

• *vectorstare* — ansamblul informațiilor necesare unui procesor pentru a controla execuția procesului; în momentul realocării procesorului componenta conservă informațiile necesare reluării ulterioare a procesului;

• *prioritate* — întreg utilizat ca măsură în departajarea cererilor simultane de resurse sau în planificarea procesorului;

• *program* — referință la procedura reprezentînd ansamblul de acțiuni și informații implicate în prelucrarea caracteristică procesului.

Utilizăm notația

prel: proces

pentru a declara că *prel* reprezintă un descriptor de proces.

În afara componentelor menționate (suficiente pentru scopul lucrării de față), un descriptor poate conține și alte elemente, cum ar fi: lista proceselor înrudite (părinte și descendenți imediați), lista capabilităților procesului, informații de cont etc. Descriptorul caracterizează comportarea dinamică a procesului și evidențiază structurile ierarhice în care acesta se încadrează, conform diferitelor criterii posibile. Astfel, componentele *vectorstare* și *program* precizează starea prelucrării curente, generată de proces plecînd de la o stare inițială dată. Componenta *prioritate* precizează nivelul pe care se situează procesul în cadrul unei ierarhii de procese stabilite pe baza importanței momentane a acestora în cadrul sistemului. Prioritatea poate fi modificată în timpul evoluției procesului, modificările urmărind de regulă realizarea unui echilibru, fie din punctul de vedere al încărcării resurselor, fie din cel al repartizării resurselor comune pe procese.

2.3.3. DESCRIEREA OPERAȚIILOR PRIMITIVE ALE PROCESELOR

Ideea de a realiza sincronizarea și comunicarea între procesele unui sistem prin intermediul unor operații primitive nu este nouă [D—1]. Astfel „P“ și „V“, „lock“ și „unlock“, „enq“ și „deq“, „wait“ și „post“, „block“ și „wakeup“, „send“ și „receive“ sînt doar cîteva exemple de operații primitive utilizate în modelele sistemelor de calcul sau în implementările lor. Aceste perechi de operații au capacitatea de a descrie două tipuri de sincronizări între procese, și anume:

• excluderea mutuală, necesară cînd două sau mai multe procese împart o resursă comună, dar numai un proces poate modifica și/sau testa starea resursei la un moment dat;

• comunicare de tip producător-consumator, care presupune că informația produsă de un proces este consumată de un alt proces.

Reprezentarea altor tipuri de interacțiuni se bazează pe utilizarea unei perechi de primitive din cele menționate și reducerea funcțiilor care trebuie descrise la aceste primitive. O altă soluție posibilă este descrierea interacțiunilor în termenii unor proceduri tipice, considerate ca operații indivizibile. Asocierea procedurilor cu resursele comune la care se referă în cadrul unei structuri mixte de date și control facilitează verificarea corectitudinii acțiunilor diverselor procese asupra resurselor comune, precum și mai buna înțelegere a interacțiunilor între procese [H—3].

Tuturor operațiilor menționate le este comun un anumit mecanism de modificare a stării proceselor: un proces se poate bloca sau poate fi activat, după cum starea resurselor comune împiedică execuția procesului ori permite reluarea ei. Aceasta conferă un caracter particular interacțiunii între procese, un proces neputînd influența direct evoluția altuia. Utilizarea resurselor comune apare ca un drept pe care un proces îl poate cîștiga, dar odată cîștigat rămîne procesului pînă cînd el însuși îl cedează celorlalte procese din sistem. În sistemele în care sincronizarea proceselor este descrisă prin operații ca cele menționate nu este permisă preluarea forțată a resurselor unui proces de către alt proces. De multe ori îmbunătățirea performanțelor unui sistem de calcul impune ca resursele să nu fie cedate proceselor pe durată nelimitată și, deci, preluarea forțată a acestor resurse de la procesele care le dețin să fie posibilă. Abordarea sistematică a proiectării sistemului de operare, confruntat cu problema alocării optime a resurselor, impune definirea și implementarea unor operații primitive care să realizeze atît interacțiunile indirecte la nivelul resurselor, cît și interacțiunile directe, explicite (acțiuni de suspendare, trezire).

Operațiile prezentate în continuare sînt definite pe descriptorii proceselor și realizează blocarea, activarea, suspendarea, trezirea și actualizarea priorității proceselor. Ele sînt descrise ca o colecție de proceduri ce operează pe resursele comune reprezentate de descriptorii proceselor și ai procesoarelor. Pentru ca efectele lor asupra datelor comune să nu fie întîmplătoare execuțiile procedurilor se exclud în timp.

Operația de blocare are loc numai prin acțiunile procesului. De aceea blocarea poate surveni numai cînd procesul este în execuție, ca urmare a apelului procedurii *blochează*. Notînd cu * procesul care apelează procedura, operația de blocare corespunde algoritmului 2.10.

Algoritmul 2.10

```
procedure blochează;
  var cpu : procesor;
  begin
    *.stare:='blocat';
    cpu:=*.procesorex;
    cpu.procex:=c;
    șterge (*, cpu. lpa);
    cpu.planificator
  end
```

Pentru ca execuția planificatorului să se soldeze mereu cu ocuparea procesorului considerăm că pentru fiecare procesor există un proces ciclic inefictiv (de joasă prioritate), întotdeauna pregătit, denumit *c*. Dacă în *lpa* nu există un alt proces pregătit, acestuia i se va aloca procesorul prin execuția planificatorului. Procedura „șterge" realizează ștergerea procesului * din lista *cpu.lpa*.

Operația de activare a unui proces este declanșată prin acțiunile executate de alt proces. Ca parametru al procedurii de activare se specifică procesul care este activat (algoritmul 2.11).

Algoritmul 2.11

```
procedure activează (proc:proces);
    var cpu: procesor;
    begin cpu:=proc. procesorex;
        inserează (proc, cpu. lpa);
        if proc. stare='blocatsuspendat' then
            proc. stare:='suspendat'
        else if proc. stare='blocat' then
            begin proc. stare:= 'pregătit'
                cpu. planificator
            end
    end
```

Ca urmare a acestei operații procesul activat este introdus în „*lpa*". Dacă starea procesului devine 'pregătit', este reconsiderată alocarea procesorului prin execuția planificatorului.

Operația de suspendare are ca parametru procesul care este suspendat (algoritmul 2.12).

Algoritmul 2.12

```
procedure suspendă (proc: proces);
var cpu : procesor;
begin
    if proc. stare='blocat' then
        proc. stare:='blocatsuspendat'
    else if proc. stare='pregătit' then
        proc. stare='suspendat'
    else if proc. stare='execuție' then
        begin proc. stare:='suspendat';
            cpu:=proc.procesorex;
            cpu. procex:=c;
            cpu.planificator
        end
end
```

Efectul operației este modificarea stării procesului suspendat. Cînd suspendarea se referă la un proces în execuție atunci este reconsiderată alocarea procesorului corespunzător.

Operația de trezire are ca parametru procesul care este trezit. Dacă în urma execuției operației starea procesului devine 'pregătit' este replanificat procesorul (algoritmul 2.13).

Algoritmul 2.13

```
procedura trezește (proc: proces);
    var cpu: procesor;
    begin
        if proc. stare = 'blocatsuspendat' then
            proc. stare: = 'blocat'
```

```
        else if proc. stare = 'suspendat' then
                begin proc. stare: = 'pregătit';
                  cpu: = proc. procesorex;
                  cpu. planificator
                end
end
```

Operațiile de suspendare și de trezire modelează intervenția directă a unui proces în evoluția altui proces din sistem. Procesul suspendat nu mai intră în competiție cu celelalte procese pentru alocarea procesorului, fiind ignorat de planificator. Prin operațiile de suspendare și de trezire procesul poate menține controlul execuției altui proces din sistem. Această facilitate este utilă, de exemplu, într-un sistem în care unul din procese este depanat sau executat în regim interactiv. Execuția unui astfel de proces este suspendată în mod repetat, în vederea examinării stării prelucrării sau a unor eventuale modificări ale acestei stări.

Procesul suspendat rămîne cuplat la lista în care se afla în acel moment: dacă era în execuție ori pregătit se oprește în *lpa*; în situația că se găsea în coada de așteptare asociată unei resurse rămîne în acea coadă. Nici trezirea procesului nu provoacă scoaterea lui din lista în care se află în acel moment.

Operația de stabilire a priorității poate fi efectuată în scopul îmbunătățirii unor performanțe ale sistemului. Prioritatea unui proces este o entitate dinamică, a cărei valoare este posibil să fie redefinită pe parcursul evoluției procesului. Parametrii procedurii de stabilire a priorității sînt: procesul a cărui prioritate este redefinită și noua valoare a priorității (algoritmul 2.14).

Algoritmul 2.14.

```
procedure setprioritate (proc: proces, pri: integer);
    var cpu: procesor;
    begin proc. prioritate:=pri;
        cpu: = proc. procesorex;
        if proc. stare = 'pregătit' or
           proc. stare = 'execuție' then
            begin șterge (proc, cpu. lpa);
                inserează (proc, cpu. lpa);
                cpu. planificator
            end
    end
```

2.3.4. PLANIFICATORUL

Operațiile primitive de blocare, activare, suspendare, trezire și de setare a priorității modifică valorile unor componente ale descriptorilor proceselor, ceea ce face uneori necesară realocarea procesorului. Funcția de alocare este îndeplinită de planificatorul procesorului, care este pus în execuție ca o procedură în operațiile primitive menționate. Concret, sarcina acestuia este de a selecta, conform unui anumit criteriu, unul din procesele din *lpa* și de a-l trece în execuție. În varianta prezentată în algoritmul 2.15 redat în continuare criteriul de selecție este prioritatea proceselor, planificatorul menținînd

în execuție procesele pregătite cele mai prioritare din *lpa*. Planificatorul gestionează n procesoare, care au acces la o memorie comună (algoritmul 2.15).

Algoritmul 2.15

```
procedure planificator;
var proc, procesînexecuție : proces, prcm: procesor;
begin
   * găsește procesul pregătit cel mai prioritar (proc);
   * găsește procesorul cu procesul avînd prioritate minimă (prcm);
   procesînexecuție: = prcm. procex;
   if proc. prioritate > procesînexecuție. prioritate then
      begin procesînexecuție. stare: = 'pregătit';
            proc. stare: = 'execuție';
            prcm. procex: = proc
      end
end
```

Deoarece prin execuția planificatorului se poate schimba procesul în execuție, implementarea operațiilor primitive trebuie să asigure actualizarea corectă a componentei *vectorstare* a procesorului și salvarea vechiului conținut al acesteia în descriptorul procesului înlocuit.

Algoritmul cuprinde două operații nestandard (marcate cu *). Ele sînt dependente de modul de organizare a listei *lpa*, respectiv de felul în care planificatorul ține evidența procesoarelor gestionate.

2.3.5. UTILIZAREA PRIMITIVELOR DE SINCRONIZARE PENTRU COMUNICAREA ÎNTRE PROCESE

Pe baza interacțiunilor primitive între procese este posibil să fie descrise unele mai complicate. Se convine să se reprezinte un anumit tip de interacțiune sub formă de structură abstractă de date, numită *resursă compusă*, care precizează atît datele comune proceselor ce interacționează, cît și operațiile (descrise în forma unor proceduri) posibil de efectuat cu aceste date. Menționăm că un proces oarecare are acces la datele comune doar prin execuția unei proceduri din resursa compusă respectivă. În plus, procedurile resursei compuse constituie o regiune critică, la un moment dat un singur proces putînd executa o singură operație a resursei. Se evită astfel utilizarea haotică a datelor comune, care ar putea conduce la erori greu de depistat. Pe parcursul execuției unei proceduri, există posibilitatea ca un proces să se blocheze. Blocarea este însoțită de eliberarea regiunii critice corespunzătoare, permițîndu-se astfel unui alt proces să efectueze o procedură a resursei compuse. Activarea ulterioară a procesului blocat se face numai din interiorul unei proceduri locale aceleiași resurse compuse. Descrierea resursei compuse presupune precizarea variabilelor comune grupate și prezentarea procedurilor resursei.

Convențiile menționate vor fi utilizate în scopul descrierii interacțiunii proceselor unui sistem de calcul prin schimbul de informații sub formă de mesaje: un proces „producător" elaborează un mesaj și îl transmite unui proces „consumator", care îl recepționează și îl prelucrează. Pentru a evita

desincronizările posibile, cauzate de vitezele de lucru diferite ale proceselor, definim resursa compusă „mesager", constînd dintr-o listă de mesaje disponibile, o listă de cereri și două operații asociate, astfel că:

- mesajele eliberate de procese se introduc în lista de disponibilități;
- un element al listei de cereri are două componente — procesul care a făcut cererea de mesaj și referința la structura ce va conține mesajul în momentul alocării;
- cele două operații definite — „cere" și „eliberează" — pun în corespondență cererile de mesaje ale unor procese, cu mesajele eliberate (produse) de alte procese.

Cerințele impuse resursei sînt următoarele:

(*a*) numărul de mesaje transmise proceselor consumatoare nu poate depăși pe al celor eliberate de procesele producătoare; mesajele produse, dar neconsumate, sînt păstrate în lista disponibilităților, astfel că nu se pierde nici un mesaj;

(*b*) procesele ale căror cereri nu pot fi satisfăcute sînt înregistrate în lista de cereri; la producerea următoarelor mesaje, ele sînt transmise proceselor solicitante din această listă;

(*c*) mesajele sînt transmise în aceeași ordine în care sînt eliberate.

Descrierea resursei este dată în algoritmul 2.16.

Algoritmul 2.16

```
mesager: resursă-compusă;
    var
        disponibil: listă-de-mesaje;
        cereri: listă-de-perechi (proces, ↑ mesaj);
procedure cere (var mes: ↑ mesaj);
    begin
        if există (disponibil) then
            extragemesaj (mes, disponibil)
        else
            begin inserează ((*, mes), cereri);
                blochează
            end
    end;
procedure eliberează (msj: mesaj);
    var proc: proces; mes: ↑ mesaj;
    begin
        inserează mesaj (msj, disponibil);
        if există (cereri) then
            begin extrage ((proc, mes), cereri);
                extragemesaj (mes, disponibil);
                activează (proc)
            end
    end
```

Pentru urmărirea ușoară a algoritmului 2.16 se fac precizările:

1) Datele comunicate între procese pot avea structuri foarte diverse. Deoarece detaliile de organizare a acestora nu sînt importante, considerăm că ele se încadrează într-un tip neprecizat denumit mesaj.

2) Cele două liste utilizate au organizare FIFO.

3) Notația „↑ mesaj" indică referința unui mesaj.

4) Funcția „există" are valoarea TRUE dacă în lista dată ca parametru există cel puțin un element.

5) Procedura „inserează̆mesaj" inserează un mesaj în lista de mesaje; procedura „extragemesaj" extrage un mesaj în zona indicată de parametrul *mes*.

6) Procedurile „inserează" și „extrage" manevrează perechi de elemente de forma (proces, mesaj).

7) Notația * este folosită pentru a desemna procesul executant al procedurii.

Procesul care execută o operație „cere" specifică în apel zona ce urmează să conțină mesajul. Dacă există un mesaj disponibil, va fi plasat în zona indicată și operația se termină. Altfel, procesul apelant și referința zonei indicată de el sînt înregistrate în lista de cereri, iar procesul se blochează.

Procesul care execută o operație „eliberează" specifică în apel mesajul transmis. Mesajul este înregistrat în lista de disponibilități, după care sînt testate solicitările existente. Dacă există o solicitare (cel puțin) se face servirea ei și activarea procesului solicitant.

Una din căile posibile de a arăta că procedurile unei resurse compuse sînt corecte constă în a demonstra că o anumită aserțiune despre valorile variabilelor comune este adevărată înainte de execuția operațiilor asociate și rămîne adevărată după efectuarea oricăreia dintre ele. O astfel de aserțiune se numește *invariant*.

Pentru demonstrarea corectitudinii soluției propuse se fac următoarele notații: *ne*- numărul de execuții ale operației „eliberează"; *-nc* — numărul de execuții ale operației „cere"; *-nd* — numărul de mesaje disponibile; *-nl* — numărul de elemente din lista de cereri; *-nr* — numărul de mesaje consumate (recepționate).

Cerința (*c*) de corectitudine este satisfăcută dacă lista de mesaje are organizare FIFO. Corespunzător condițiilor (*a*) și (*b*), invariantul este:

$$(1)\quad nd = ne - nr$$

și

$$(2)\quad nl = nc - nr$$

Să demonstrăm că el este satisfăcut atît inițial, cît și după execuția oricăreia din cele două operații. Inițial, valorile variabilelor enumerate anterior sînt zero, astfel că (*1*) și (*2*) sînt verificate.

Execuția unei operații „cere" provoacă mărirea lui *nc* cu o unitate. Dacă există un mesaj disponibil, el este extras (*nd* scade cu o unitate) și transmis procesului solicitant (*nr* crește cu o unitate). În situația că nu există mesaje disponibile procesul solicitant este introdus în lista de cereri (deci *nl* crește cu o unitate). Ca urmare, se obțin pentru relațiile (*1*) și (*2*) fie $(nd-1)=ne-(nr+1)$ și $nl=(nc+1)-(nr+1)$, fie $nd=ne-nr$ și $(nl+1)=(nc+1)-nr$; așadar invariantul este verificat după o operație „cere".

Execuția unei operații „eliberează" determină mărirea lui *ne* cu o unitate. Dacă există cereri de mesaje este transmis un mesaj (*nr* crește cu o unitate și *nl* scade cu o unitate). Altfel mesajul este introdus în lista de disponibilități (*nd* crește cu o unitate). Ca urmare avem fie $nd=(ne+1)-(nr+1)$ și $(nl-1)=nc-(nr+1)$, fie $(nd+1)=(ne+1)-nr$ și $nl=nc-nr$, astfel că invariantul este verificat după o operație „eliberează". Cu aceasta, demonstrația invarianței relațiilor (*1*) și (*2*) este completă.

2.4. PROIECTAREA PROTOCOALELOR PENTRU SISTEME TELEINFORMATICE

Proiectarea unui sistem teleinformatic poate fi mult simplificată prin împărțirea funcțiilor sale în niveluri și limitarea domeniului considerat la un moment dat în proiectare la un singur nivel funcțional. Pentru un nivel oarecare datele inițiale de proiectare se prezintă în forma unor specificații referitoare atît la serviciile pe care trebuie să le realizeze nivelul proiectat, cît și la serviciile utilizate de el din nivelurile inferioare. Proiectarea conduce la stabilirea obiectelor (proceselor) componente ale nivelului și a protocolului de cooperare prin care se realizează serviciile specificate. Descrierea unui astfel de protocol se poate face în limbaj natural, dar utilizarea unor modele formale de descriere oferă procesului de proiectare numeroase avantaje: evită ambiguitățile, permite verificarea protocolului (a proiectului), oferă posibilitatea unor implementări diferite dar cu aceeași funcționalitate etc.

Modelele algoritmice reprezintă un instrument important de exprimare într-o formă concisă și clară a protocoalelor. De altfel, acesta este și motivul care a condus la includerea conceptelor lor în limbaje de programare actuale cum sînt: Pascal concurent, Edison, Modula-2, ADA etc. [D—3]. Dar apropierea instrumentelor de modelare față de limbajele de implementare creează și unele dificultăți, aspecte importante ale proiectării pierzîndu-se în mulțimea de amănunte de implementare.

Modelele tranziționale constituie o altă categorie importantă de modele, avînd la bază reprezentarea obiectelor prin mulțimi de stări și mulțimi de tranziții ce reprezintă acțiunile obiectelor ca reacții la producerea diferitelor evenimente. Dintre ele menționăm automatele finite, rețelele Petri, rețelele UCLA etc. Dezavantajul acestor modele este că protocoalele — relativ complexe — reclamă un număr mare de stări și de tranziții, fiind necesare metodologii speciale pentru simplificarea și tratarea lor etapizată.

În fine, *modelele hibride* au la bază modele tranziționale, la care se adaugă module — program ce detaliază acțiunile obiectelor pentru fiecare tranziție în parte. O tratare sistematică a modelelor tranziționale și hibride poate fi găsită în lucrarea [P—1].

Modelele algoritmice — deși sînt mai apropiate de implementările protocoalelor — ascund, așa cum s-a menționat, unele aspecte importante legate de cooperarea între procese, în spatele construcțiilor sintactice alese în scopul exprimării foarte concise a comportării obiectelor descrise. Aceasta a determinat utilizarea cu preponderență a modelelor tranziționale și hibride în fazele incipiente ale proiectării, ca și pentru validarea și simularea protocoalelor. Majoritatea instrumentelor de validare prin verificare automată, testare sau simulare cunoscute actualmente au la bază modele tranziționale ori hibride. Au fost desigur înregistrate progrese și în utilizarea modelelor algoritmice, îndeosebi în ceea ce privește validarea prin demonstrarea corectitudinii, neajungîndu-se însă la succese notabile în automatizarea acestui proces.

Structura unui sistem de proiectare și implementare asistate de calculator a protocoalelor este prezentată în figura 2.10. Sistemul acceptă la intrare descrieri de protocoale într-un limbaj, de obicei, specializat. Pentru orice model, compilatorul sistemului produce o formă intermediară care conține principalele informații necesare validării, simulării sau implementării protocolului. Pe baza acestei forme intermediare și a stării inițiale a modelului

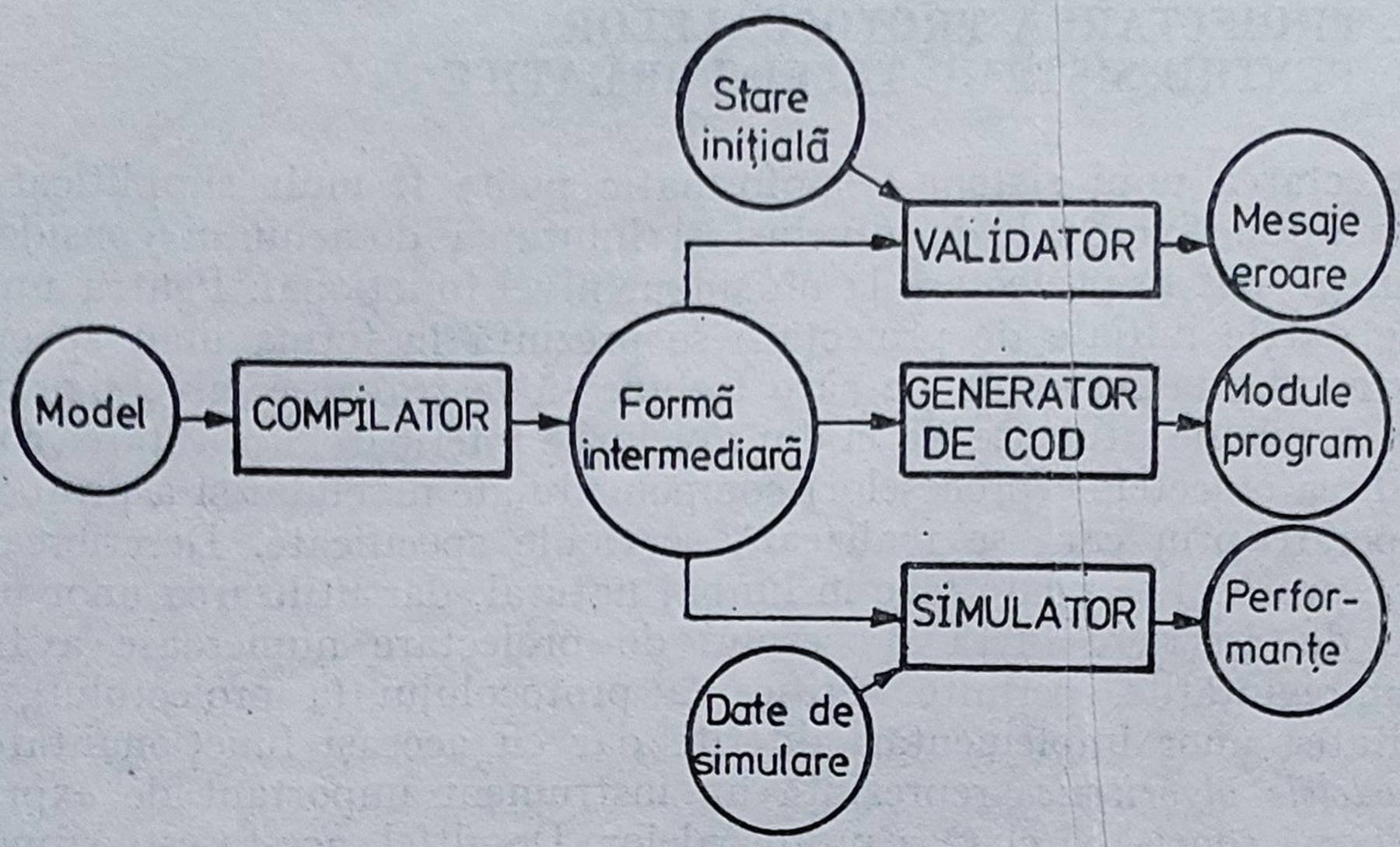


Fig. 2.10. Structura unui sistem de proiectare asistată de calculator a protocoalelor.

furnizate de proiectantul protocolului, validatorul analizează corectitudinea modelului. Metoda de verificare cea mai des utilizată în prezent se bazează pe generarea mulțimii stărilor realizabile din starea inițială dată. În acest scop, pentru fiecare stare realizabilă sînt selectate tranzițiile executabile și determinate stările succesoare. În vederea evitării duplicatelor, se memorează toate stările succesoare distincte obținute. Procesul se oprește în momentul în care nu se mai pot adăuga stări noi la cele deja obținute.

În timpul procesului de generare a spațiului stărilor realizabile pot fi detectate diferite erori de proiectare a protocolului analizat, cum ar fi [Ș—1]: blocarea definitivă, indicată de prezența unei stări realizabile fără succesor, diferită de stările finale cunoscute ale modelului; proiectarea incompletă sau imposibilitatea ca un proces să recepționeze mesajele transmise de un alt proces; depășirea capacității canalelor; ciclarea indefinită etc. La detectarea unei erori, validatorul pune la dispoziția proiectantului informații suplimentare, care înlesnesc stabilirea cauzei și a corecțiilor necesare.

Pentru analiza performanțelor prin simulare trebuie furnizate simulatorului o serie de date referitoare nu numai la starea inițială, ci și la duratele diferitelor tranziții (acțiuni) ale proceselor, la prioritățile relative ale acestora etc. De obicei simularea este condusă prin evenimente, desfășurîndu-se astfel: simulatorul păstrează în permanență o listă de evenimente posibile împreună cu momentele producerii lor. El alege evenimentul cu timpul de producere minim și execută acțiunea declanșată. Ca urmare, lista evenimentelor posibile se modifică în urma adăugării de noi evenimente ori a invalidării unora deja înregistrate, după care revine la selectarea unui nou eveniment. Pe parcursul simulării sînt înregistrate mărimile cantitative necesare calculului timpilor de răspuns, a întîrzierilor înregistrate în transmiterea mesajelor, a gradului de ocupare a resurselor utilizate etc. Prin modificarea datelor de simulare se poate realiza studiul parametrizat al unui model, în vederea alegerii variantei mai bune. De asemenea, simulatorul semnalează unele erori de funcționare a protocolului dacă simularea nu a fost precedată de validarea protocolului.

Modelul — după ce a fost corectat și îmbunătățit în urma validării și a simulării — poate fi folosit ca bază pentru implementarea protocolului. În general, implementarea unui protocol corespunde ca structură modelului său, fiecare proces cuprinzînd următoarele elemente: tabela de tranziții ce indică stările procesului, stările succesoare și condițiile de execuție a tranzițiilor; modulele — program de acțiune asociate tranzițiilor; modulele — program de interfață cu alte niveluri; modulul — dispecer care, la producerea unui eveniment, evaluează condițiile de producere a tranzițiilor, pune în execuție modulul de acțiune corespunzător și produce modificările necesare în starea curentă a procesului. Generatorul de cod (fig. 2.10) poate servi pentru producerea automată a tabelei de tranziții, a modulelor de acțiune și a modulului dispecer, în forma unor module-program și proceduri de bibliotecă. Întregul protocol este implementat prin adăugarea modulelor suplimentare ce încapsulează elementele dependente de implementare și care trebuie — așadar — codificate separat de către proiectant.

CAPITOLUL 3

UTILIZAREA MODELELOR CU ŞIRURI DE AŞTEPTARE ÎN PROIECTAREA STI

Sînt prezentate o serie de metode şi tehnici utile în calculul şi în predicţia performanţelor sistemelor teleinformatice. În proiectarea STI şi în evaluarea performanţelor acestor sisteme, instrumentele cele mai frecvent folosite sînt simularea aleatoare a evenimentelor discrete şi teoria şirurilor de aşteptare. Capitolul abordează succint o problematică diversă, cuprinzînd modele cu şiruri de aşteptare, simulare, măsurarea şi analiza performanţelor, dînd o amploare mai largă în finalul capitolului exemplelor de aplicare a metodelor cu şiruri de aşteptare în proiectarea STI.

3.1. MODELE CU ŞIRURI DE AŞTEPTARE

3.1.1. MODEL. MODELARE. SIMULARE

Prin noţiunea de model se înţelege descrierea abstractă a unui sistem cu ajutorul căruia se aproximează realitatea obiectivă pe baza observaţiilor şi măsurătorilor. Abaterile modelului faţă de realitate depind de stadiul de cunoaştere al sistemului şi sînt puse în evidenţă în urma simulării sau a măsurării. Modelarea constituie o metodă de studiu a proceselor sau fenomenelor prin substituţia obiectului real de cercetat cu un sistem uşor de studiat şi mînuit, ce are un grad mai mic ori mai mare de simplificare. Modelele se împart în două categorii: deterministe şi stohastice, respectiv vag definite. La cele deterministe — mărimile ce descriu procesul modelat sînt determinate de relaţii de tip funcţie, iar la modelele stohastice — elementele definitorii ale modelului sînt variabile aleatoare, cu distribuţie cunoscută sau determinabilă. Pentru modelele vag definite — variabilele nu se pot caracteriza nici măcar în probabilitate, cunoscîndu-se doar plaja de valori în care ele se află.

Noţiunea de simulare semnifică o imitare a comportării sistemului studiat (D-4, R-1), reprezentat prin modelul său, existenţa variabilelor aleatoare în model impunînd experimentul repetat, cu caracter statistic. Simularea nu oferă o soluţie optimă pentru problemele de decizie, ci un tablou al rezultatelor ce s-ar obţine în timp şi spaţiu ca urmare a folosirii variantelor utilizate. Metoda simulării — caracterizată printr-un anumit grad de incertitudine

relativ la plaja de variație a unor parametri și la natura acestora — este suficient de eficientă în aplicarea tehnicilor de căutare de tip *trial and error* (încercare-eroare) în vederea determinării soluțiilor satisfăcătoare (de obicei suboptime). În timp ce modelarea matematică conduce la modele ce oferă posibilitatea să fie studiate analitic — rezultînd variante de decizie, cu o exactitate mai mare sau mai mică, în funcție de completitudinea modelului și de exactitatea metodelor de rezolvare —, simularea permite obținerea unei imagini ce reprezintă prelungirea în timp și/sau spațiu a desfășurării unui proces ale cărui coordonate definitorii se consideră cunoscute statistic din trecut și care se presupune a funcționa aproape întotdeauna în regim staționar, adică fără modificări substanțiale ale caracteristicilor sale de bază. Aceste aserțiuni sînt valabile pentru procese dinamice, în cazul proceselor statice simularea fiind folosită ca metodă de calcul (simularea Monte-Carlo) atunci cînd calculul analitic nu este posibil sau este costisitor. Modelul de așteptare constituie o descriere formală a unui sistem de așteptare, furnizînd informații despre modul de intrare a cererilor în sistem, structura șirurilor de așteptare și a stației de servire, precum și disciplina de rezervare. Sistemul de așteptare nu ilustrează decît aproximativ funcționarea sistemului real, întrucît intrarea cererilor în sistem este pur aleatoare, iar aparatul matematic care să descrie complet procesul de servire nu există.

Teoria șirurilor de așteptare dă posibilitatea calculării valorii unor mărimi precum timpul mediu de așteptare, lungimea medie a șirurilor, „timpul mort“ * petrecut în sistem, timpul de serviciu etc. Deoarece procesele de așteptare în STI sînt de o complexitate deosebită, modele matematice corespunzătoare nu pot fi plenar aplicate, fiind posibilă doar obținerea estimărilor dorite prin simularea întregului sistem sub diferite sarcini de lucru. Se estimează astfel valorile și distribuțiile de probabilități asociate mărimilor menționate mai sus. În unele condiții speciale dispersiile și valorile medii se consideră ca măsurători ale performanței sistemului.

3.1.2. LEGE EXPONENȚIALĂ. DISTRIBUȚII FRECVENTE ÎN TEORIA AȘTEPTĂRII

Unele modele matematice sau de simulare utilizate la studierea performanțelor și a caracteristicilor traficului într-un STI folosesc legea exponențială și procesul Poisson pentru reprezentarea timpilor între sosirile succesive ale mesajelor într-un nod ori într-un comutator sau pentru specificarea detaliilor transmisiei.

Fie cel mai simplu șir de așteptare. Mai mulți „clienți“, care poartă numerele 1, 2, 3, ..., sosesc la momentele $t_1 < t_2 < t_3 < \ldots$ în șirul de așteptare. Stația de servire tratează clienții într-o ordine oarecare. Un astfel de sistem de așteptare este specificat complet dacă se cunosc: distribuția sosirilor (a intervalului de timp între sosiri ca variabilă aleatoare sau deterministă); distribuția duratei servirii (variabilă aleatoare ori deterministă); disciplina de servire (modul de satisfacere a cererilor); capacitatea sistemului (numărul tranzacțiilor ce se pot afla simultan în sistemul de servire — numărul tranzacțiilor care așteaptă să fie servite + numărul celor în curs de servire).

---

* Perioada liberă a stațiilor de servire

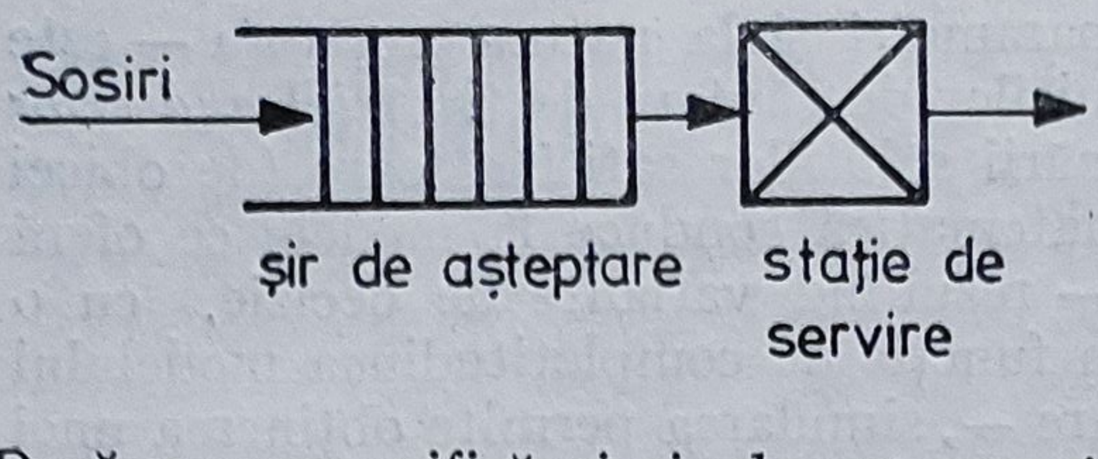


Fig. 3.1. Șir de așteptare simplu

Dacă nu se specifică nimic despre capacitatea stației, ea se consideră infinită. Tranzacția reprezintă o cerere de servire ce poate proveni de la una sau mai multe surse. Stația de servire are unul ori mai multe posturi de servire.

Se presupune că timpii $I_1 = t_1 - 0$, $I_2 = t_2 - t_1$, ... sînt variabile aleatoare, adică cantități ale căror valori exacte sînt necunoscute, dar pentru care se poate defini o distribuție de probabilitate. Variabilele S_1, S_2, S_3, ..., reprezentînd timpi de serviciu succesivi, sînt de asemenea variabile aleatoare, iar cei între sosirile succesive I_1, I_2, I_3, ..., sînt distribuiți în mod identic de legea exponențială:

$$P(I_j < x) = 1 - e^{-\lambda x}, \ (\forall)\ j \geqslant 1. \tag{3.1}$$

Aceeași ipoteză fiind făcută asupra timpilor de serviciu, se obține:

$$P(S_i < x) = 1 - e^{-\mu x}, \ (\forall)\ i \geqslant 1. \tag{3.2}$$

Parametrii λ și μ ai legilor exponențiale sînt reali pozitivi și finiți, iar intersosirile și timpii de serviciu sînt variabile independente. Dacă cererile de serviciu sosesc independent una de cealaltă și probabilitatea ca o cerere să sosească într-un interval Δt este proporțională cu mărimea acestui interval, atunci caracterizarea acestui proces o dă *repartiția Poisson*. Constanta de proporționalitate λ este chiar numărul mediu de sosiri în unitatea de timp. Întrucît probabilitatea ca o cerere să sosească în intervalul Δt este $\lambda \cdot \Delta t$ se poate demonstra că densitatea de probabilitate pentru acest tip de sosiri este exponențial negativă:

$$f(t) = e^{-\lambda t} \ (t > 0)$$

și că probabilitatea ca în intervalul de lungime t să sosească n cereri este de:

$$P(n) = \frac{(\lambda t)^n \cdot e^{-\lambda t}}{n!} \tag{3.3}$$

În mod similar, cînd durata servirii este o variabilă aleatoare numărul cererilor servite într-un interval Δt, respectă o aceeași lege Poisson de parametru μ (numărul mediu al cererilor servite în unitatea de timp), cu o repartiție exponențială pentru durata servirii. În practică procesele Poisson sînt observate în sistemele reale atunci cînd sosirile provin de la un număr mare de surse independente. Acesta este cazul sosirii unor mesaje la un calculator dacă mesajele sînt emise de un mare număr de terminale dispersate geografic și utilizate de operatori independenți între ei.

Pentru traficul telefonic este specifică *distribuția Erlang*. Densitatea de probabilitate pentru intervalul de timp intersosiri este următoarea [G — 1]:

$$f(t) = (\lambda t)^k \cdot \frac{e^{-\lambda k t}}{(k-1)!} \cdot t^{k-1}, \qquad k \in Z_+^*. \tag{3.4}$$

În cazul cînd $k = 1$ se obține distribuția exponențială a unui sistem cu o singură unitate de servire și cu durata servirii avînd distribuția exponențială în jurul unei valori medii. Dacă $k > 1$, valorile corespunzătoare ale intervalului dintre cererile sosite în jurul valorii medii sînt grupate mai strîns, astfel încît pentru $k = \infty$ distribuția corespunde sosirilor la intervale constante.

Distribuția normală caracterizează acea variabilă aleatoare cu valorile distribuite simetric în jurul valorii medii, fiind complet descrisă prin valoarea medie m și abaterea medie pătratică σ. Funcția densitate de probabilitate are forma:

$$f(x-m) = \frac{1}{\sqrt{2\pi}} \cdot e^{-\left(\frac{x-n}{\sqrt{2\sigma^2}}\right)^2} \qquad (3.5)$$

Timpii de așteptare, timpii inter-sosiri și timpii de serviciu în STI vor fi considerați întotdeauna ca variabile aleatoare continue (v.a.c.) nenegative, iar numărul sosirilor sau plecărilor într-un interval de timp dat și lungimea șirurilor sînt variabile aleatoare discrete (v.a.d.). Unele procese întîlnite în teoria șirurilor de așteptare sînt procese stohastice, fiind funcție de timpul t și de starea inițială $t_0 = 0$. O subgrupă importantă de asemenea procese stohastice o reprezintă *procesele Markov*. Un șir de așteptare constituie un proces Markov dacă starea prezentă a sistemului la momentul t este suficientă pentru a prezice starea următoare (de exemplu, numărul de pachete sosite de pe linie) fără a se cunoaște stările sistemului în momentele anterioare lui t. Procesele Markov sînt continue în timp. Cînd sistemul modelat este studiat la momente de timp discrete t_j ($j = 0,1, ..., m, ...$) mulțimea probabilităților de stare $P_m(j)$ corespunzătoare stărilor $ST_m(m = 0, 1, 2, ..., i, ...)$ constituie un lanț Markov. De exemplu, lungimile șirului de așteptare a sosirii mesajelor la un procesor dintr-un nod de rețea în momentul cînd sosește un nou mesaj sau pleacă un mesaj servit, poate constitui un lanț Markov.

Modelul Erlang-k este format dintr-o serie de k posturi de serviciu, identice, exponențiale cu timpii de serviciu $\frac{1}{\mu}$, un singur client parcurgînd acest model (fig. 3.2 a). Se poate concepe o generalizare a distribuției Erlang-k, în care fiecare post de servire are o rată diferită a serviciilor, μ_i, $i = 1, ..., k$, posturile fiind plasate însă paralel (fig. 3.2 a). Se obține astfel *modelul*

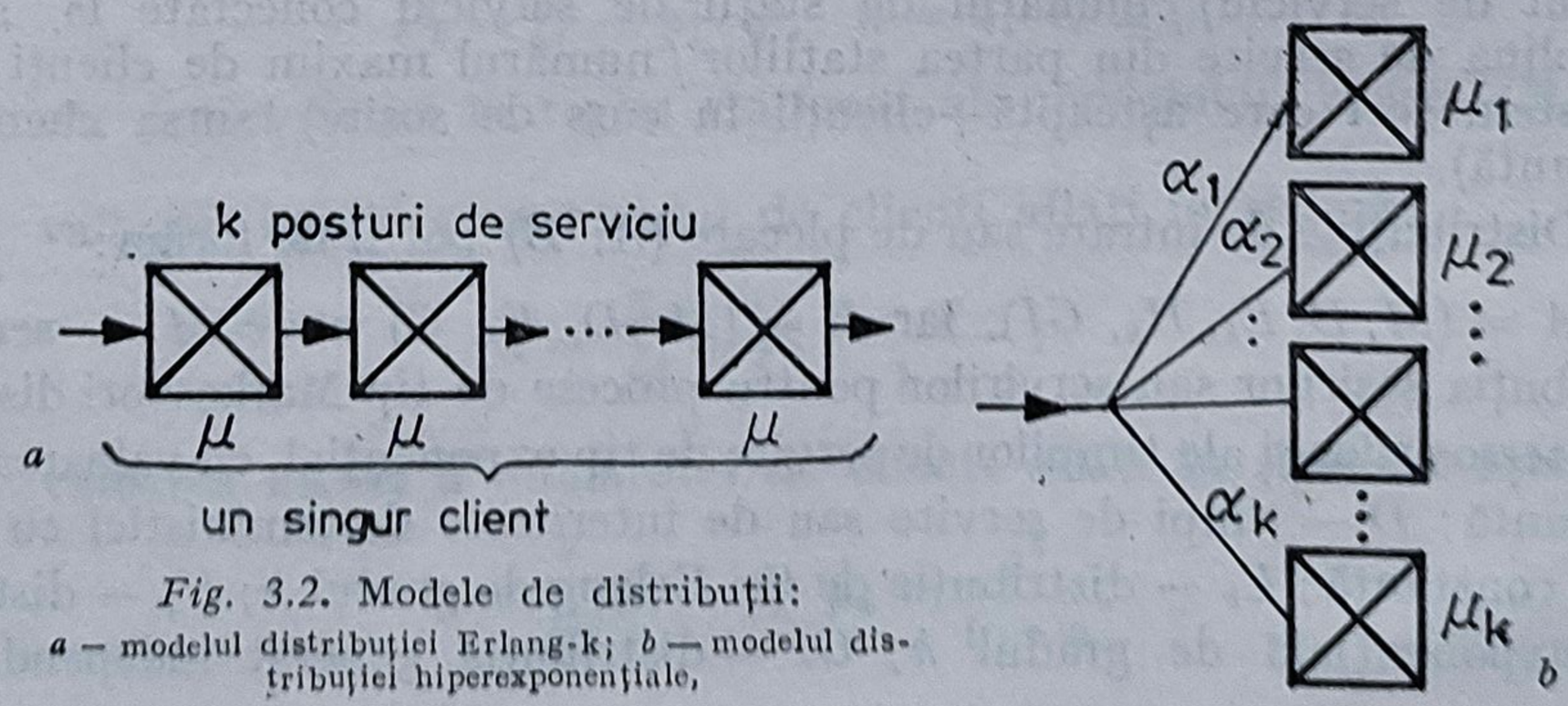


Fig. 3.2. Modele de distribuții:
a — modelul distribuției Erlang-k; *b* — modelul distribuției hiperexponențiale,

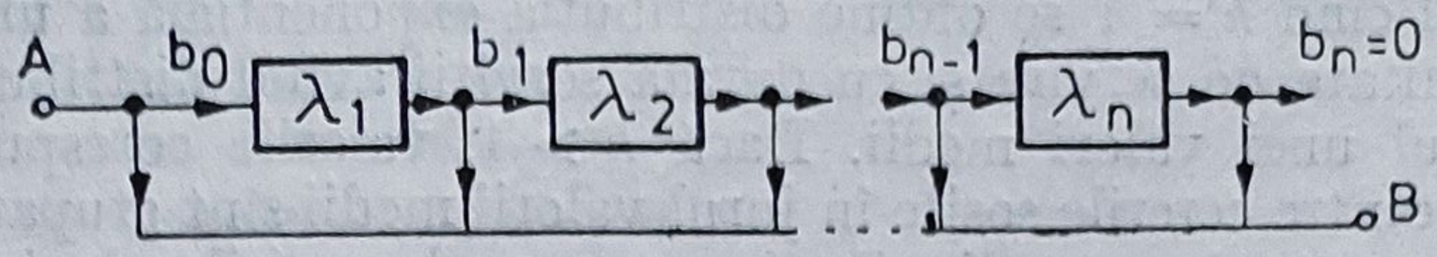


Fig. 3.3. Rețea folosită la definirea distribuției lui Cox.

hiperexponențial în care un client ce se prezintă înaintea stației va fi servit de postul i, cu probabilitatea α_i. Densitatea de probabilitate este o combinație de exponențiale negative de forma:

$$f(t) = \sum_{i=1}^{k} \alpha_i \cdot \mu_i \cdot e^{-\mu_i \cdot t}. \tag{3.6}$$

Pentru a defini o variabilă aleatoare care urmează o *distribuție Cox* se consideră rețeaua din figura 3.3, în care se presupune că între punctele A și B se poate găsi doar un singur client la un moment dat. Rețeaua este compusă din n stații cu timpi de servire conform unor legi exponențiale de parametri: λ_i, $i \in [1, n]$. Clientul care intră în rețea prin punctul A se îndreaptă fie către prima stație cu probabilitate fixă b_0, fie către B cu probabilitate $1 - b_0$. În general, ieșind din stația j el se adresează stației B cu probabilitatea $1 - b_j$ ori stației $j + 1$, cu probabilitatea b_j. Timpul total petrecut de client în rețea este o sumă de un număr aleator de variabile aleatoare exponențial distribuite. Această clasă de variabile aleatoare cu distribuție de tip Cox conține distribuții Erlang și sume de distribuții Erlang (exponențiale, hiperexponențiale).

3.1.3. NOTAȚIA KENDALL A ȘIRURILOR DE AȘTEPTARE

Pentru specificarea sintetică a caracteristicilor unui șir de așteptare se va folosi notația Kendall generalizată, sub forma:

$$(A/B/C) : (D/E/F).$$

Aceasta precizează cei 6 factori care definesc un șir de așteptare: tipul distribuției de intrare (sau intersosirile)/tipul distribuției de servire (ori timpul de serviciu) / numărul de stații de serviciu conectate la sistem / disciplina de servire din partea stațiilor / numărul maxim de clienți admiși în sistem (cei care așteaptă+clienții în curs de sosire) / sursa chemătoare (apelantă).

Distribuțiile de intrare sau de plecare (A, B) pot fi de forma:

$A = (M, D, E_k, H_k, GI)$, iar $B = (M, D, E_k, G)$ unde M — semnifică distribuția sosirilor sau servirilor pentru procese de tip Markov ori distribuții ale intersosirilor și ale timpilor de servire de tip exponențial, cu valoarea medie constantă; D — timpi de servire sau de intersosire deterministici cu distribuție constantă; E_k — distribuție de tip Erlang de gradul k; H_k — distribuție hiperexponențială de gradul k; GI — distribuția general independentă a

sosirilor sau a timpilor intersosiri; G — distribuția generală a plecărilor ori a timpilor de servire.

Stația de servire (C) este compusă din mai multe posturi de servire legate fie în serie — situație în care se folosesc rezultatele dezvoltate pentru distribuția Erlang, —, fie în paralel — caz în care se aplică metodele de calcul pentru multistații de tip exponențial.

Disciplina de servire (D) poate fi una din următoarele: FIFO — primul sosit, primul servit; LIFO — ultimul venit, primul servit; A sau FIRO — selecția aleatoare în vederea servirii; RR — servire parțială în mai multe faze, după algoritmul de tratare circulară; GD — disciplină generală de servire (include FIFO, LIFO, FIRO). Numărul maxim de clienți admiși în sistem (E) este finit sau infinit. Sursa chemătoare (F) poate fi finită ori infinită, situație în care un utilizator aflat în serviciu poate face noi cereri de servire. De exemplu, șirul de așteptare (M/E)/2) : (A/2/6) posedă intervalele între sosiri distribuite exponențial, o lege de serviciu de tip Erlang-k, 2 posturi de serviciu, disciplina de serviciu aleatoare, o capacitate maximă de 2 clienți și doar 6 utilizatori pot primi acest șir.

3.1.4. ȘIR DE AȘTEPTARE CU O STAȚIE DE SERVIRE (M/M/1) : (FIFO/∞/∞)

Există doar o singură stație de servire, fluxul sosirilor este simplu, cu parametrul λ, iar timpul de servire a unui client este o variabilă aleatoare cu distribuție exponențială de parametru μ. Se consideră că sistemul de așteptare este deschis, clienții intrînd din exterior în sistem, primind serviciul și părăsind apoi sistemul.

Se definește factorul de utilizare (intensitatea traficului):

$$\rho = \frac{\lambda}{\mu} \text{ <Erlang>}. \tag{3.7}$$

Ne interesează probabilitatea $P_k(t)$ ca în sistem să existe k clienți la momentul t. Rezolvarea acestei probleme [S — 2, K — 3] este foarte greoaie, iar soluțiile dependente de timp sînt practic imposibil de obținut sau de manipulat. Pentru cea mai mare parte a situațiilor concrete este suficientă cunoașterea soluțiilor independente de timp, numite soluții permanente ori staționare. Se obține:

$$P_k = [1 - \rho] \cdot \rho^n, \quad n = 0, 1, 2, \ldots \tag{3.8}$$

Cîteva caracteristici în regim staționar ale modelului acesta (M//M/1) sînt:

- valoarea medie a numărului de clienți aflați în sistem:

$$\bar{n} = \sum_{k=0}^{\infty} k \cdot p_k = \frac{\rho}{1 - \rho}; \tag{3.9}$$

- valoarea medie a numărului de clienți aflați în șirul de așteptare:

$$\bar{v} = \sum_{k=1}^{\infty} (k - 1) \cdot p_k = \frac{\rho^2}{1 - \rho}; \tag{3.10}$$

• funcția densitate de probabilitate a timpului total τ de așteptare a unui client (din momentul intrării în sistem, pînă în momentul ieșirii din sistem):

$$f(t) = (\mu - \lambda) \cdot e^{-(\mu-\lambda)t}, \quad t > 0, \; f(t) = 0 \text{ pentru } t \leqslant 0; \tag{3.11}$$

• valoarea medie a timpului total de așteptare:

$$\overline{\tau} = \frac{1}{\mu - \lambda} = \frac{\overline{n}}{\lambda}; \tag{3.12}$$

• dacă timpul de servire τ' al clientului are distribuția exponențială cu parametrul μ, valoarea medie a timpului $\tau - \tau'$ de așteptare în șir este:

$$M(\tau - \tau') = M(\tau) - M(\tau') = \frac{1}{\mu - \lambda} - \frac{1}{\mu} = \frac{\lambda}{\mu(\mu - \lambda)} = \frac{\overline{n}}{\mu}. \tag{3.13}$$

Pentru a permite studiul sistemelor în care legile de serviciu sau intersosirile nu sînt exponențiale se va da un rezultat general, numit *formula lui Little*. Dacă notăm cu $N(t)$ numărul de clienți din sistem, atunci numărul mediu de clienți aflați în sistem în intervalul $[a, b]$ este:

$$\overline{N} = \frac{1}{b - a} \int_b^a N(t)\, dt. \tag{3.14}$$

Presupunem de asemenea că în acest interval au loc k sosiri. Dacă d_i este momentul plecării clientului care a sosit la timpul a_{i_j}, atunci timpul mediu de răspuns al acestor clienți va fi:

$$\overline{T} = \frac{1}{k} \sum_{j=1}^{k} (d_{i_j} - a_{i_j}), \quad \text{cu} \quad d_{i_j} \leqslant b, \quad \text{cu} \quad N(b) = 0. \tag{3.15}$$

Notînd cu $\lambda(a, b) = \dfrac{k}{b - a}$ se obține formula lui Little: (3.16)

$$\lambda(a, b)\, \overline{T} = \overline{N}. \tag{3.17}$$

3.1.5. REȚEA DE ȘIRURI DE AȘTEPTARE (REȚEAUA LUI JACKSON)

Modelul de rețea a lui Jackson se referă la sisteme compuse din N stații de serviciu și dintr-un fir de așteptare de lungime nedefinită înaintea fiecărei stații. Dirijarea unui client de la o stație la alta este reprezentată de un lanț Markov, nefăcîndu-se nici o distincție între caracteristicile probabilistice ale clienților. Astfel, dacă se notează cu (Z_i), $i \geqslant 1$ suita de stații parcurse de un client oarecare, probabilitatea condiționată a lui Z_i are în acest caz proprietatea:

$$P(Z_{i+1} | Z_1, Z_2, \ldots, Z_i) = P(Z_{i+1} | Z_i) \tag{3.18}$$

pentru $(\forall)$ $i \geqslant 1$ și $Z_i \in \{1, 2, \ldots, N\}$. Cînd $Z_i = N + 1$ aceasta reprezintă plecarea către exteriorul rețelei de șiruri de așteptare. Se consideră atît *sisteme deschise* (în care clienții sosesc din exterior potrivit unor procese Poisson,

sînt serviți și apoi părăsesc rețeaua), cît și *sisteme închise* (ce au un număr constant de clienți). Exemple de sisteme închise îl constituie sistemele interactive în care buclele ce închid sistemul cuprind șirul de intrare al programelor, stația de servire UC și mulțimea terminalelor de unde utilizatorii își introduc programele. În cazul probabilistic timpii de serviciu la fiecare șir vor fi independenți între ei și distribuiți după legi exponențiale cu parametri putînd depinde de lungimea șirurilor respective. Acest model de rețea a lui Jackson are proprietatea interesantă de a poseda o soluție staționară sub formă de produs pentru probabilitățile asociate lungimilor șirurilor; soluția se scrie sub forma unui produs de distribuții marginale pentru lungimile fiecărui șir de așteptare. Din acest motiv este mult utilizat la modelarea sistemelor informatice sau teleinformatice foarte complexe.

Modelul lui Jackson corespunde unei situații reale numai cînd există certitudinea că utilizatorii de la terminale sînt suficient de prompți pentru a garanta întotdeauna că cel puțin o tranzacție, lucrare sau interogare aflată în așteptare va fi servită. O soluție a modelului rețelei închise poate fi exprimată sub forma produs a stării sistemului dată de:

$$P(S_1, S_2, \ldots, S_T) = \frac{1}{k_N} \cdot \prod_{j=1}^{T} P(S_j), \tag{3.19}$$

unde $P(S_1, S_2, \ldots, S_T)$ reprezintă probabilitatea stării n a rețelei închise a sistemului, descrisă prin modelul cu T șiruri și c stații de servire de tipul (M/M/c); (FIFO/∞/∞/). Se poate scrie:

$$n = (I_1, I_2, \ldots, I_n), \quad I_j \geqslant 0, \quad j = 1, 2, \ldots, T, \tag{3.20}$$

$I = \sum_{j=1}^{n} I_j$ este numărul total de lucrări aflate în cele $j = 1, 2, \ldots, T$ șiruri, iar $P(S_j)$ sînt factorii corespunzători probabilității stării șirului $j = 1, 2, \ldots, T$. Factorul K_N reprezintă o constantă de normalizare a rezolvării sistemului prin soluția sub formă produs. Pentru calculul ei se însumează un număr anumit de termeni în funcție de mulțimea spațiului stărilor sistemului $S(n)$. Numărul de termeni este următorul:

$$N_S = C_{I+T-1}^{T-1} = \frac{(I + T - 1)!}{(T - 1)! \cdot I!} \tag{3.22}$$

Modelul lui Jackson prezintă însă mai multe neajunsuri pentru modelarea STI: restricția ca timpii de serviciu să fie dați de o lege exponențială și ca sosirile să urmeze o lege Poisson; imposibilitatea de a avea mai multe clase de comportament statistic a clienților din rețea; utilizarea exclusivă a disciplinei FIFO pentru stațiile de servire; imposibilitatea de a ține seama de dependența între legile și disciplinele de servire și starea totală a modelului; dirijarea clienților în rețea se reprezintă printr-un lanț Markov de ordinu unu.

3.1.6. REȚELE BCMP

La începutul anilor 1970 s-au relansat cercetările de găsire a unor soluții simple pentru noile rețele de șiruri de așteptare ce modelau STI tot mai complexe și care să nu prezinte inconvenientele rețelelor de tip Jackson. În acest

timp au apărut rețelele BCMP (ale lui Baskett, Chandy, Muntz și Palacios) care păstrează soluția sub formă produs a stării de echilibru, introducînd diferite clase de clienți și noi discipline de serviciu.

În continuare vor fi abordate doar cîteva elemente despre acest tip de rețele, propunînd cititorului pentru detalierea subiectului să consulte lucrarea [G-1].

Rețelele BCMP pot avea un număr de R clase de clienți ($R \geqslant 1$). Fie N numărul stațiilor de serviciu. Dacă sistemul este deschis stațiile O și N + 1 sînt stații fictive ce reprezintă intrarea și, respectiv, ieșirea din rețea. Clienții care circulă în rețea aparțin uneia din cele R clase. Probabilitățile de dirijare a clientului prin rețea sînt date prin intermediul unui lanț Markov cu probabilitatea de tranziție:

$$P = \{p_{ir, jr'}\}, \quad \text{unde} \quad 0 \leqslant i \leqslant N, \; 1 \leqslant j \leqslant N + 1, \; r \geqslant 1, \; r' \geqslant R. \qquad (3.23)$$

Mărimea $p_{ir, jr'}$ semnifică probabilitatea ca un client de clasă r din stația i să fie dirijat către șirul j în clasa r'; $p_{ir, N+1r'}$, dă probabilitatea ca un client de clasă r din stația i să părăsească rețeaua la sfîrșitul serviciului său.

Stațiile de serviciu ale unei rețele BCMP pot să se supună la oricare din restricțiile celor patru posibilități următoare:

Tipul 1. Disciplina de serviciu FIFO, stația are un singur post de servire, iar timpul de servire este exponențial distribuit cu aceeași valoare medie pentru toate clasele de clienți, parametrul legii exponențiale putînd fi funcție de lungimea șirului.

Tipul 2. Disciplina de serviciu este cea corespunzătoare timpului partajat (time-sharing), prin care un client al stației primește un timp de serviciu de $\frac{1}{k}$ secunde atunci cînd există k clienți în stație. Această cuantă de serviciu primită la fiecare trecere prin stația de servire tinde la zero. Distribuțiile timpilor de serviciu pot fi de forma legii lui Cox, distincte pentru fiecare clasă de clienți.

Tipul 3. Stația de servire conține un număr de posturi de servire superior sau egal celui mai mare număr de clienți care vor putea să se găsească în șirul de așteptare la acea stație. Timpii de servire se supun legii lui Cox.

Tipul 4. Disciplina de serviciu este LIFO cu o prioritate absolută pentru clientul nou care sosește. Există un singur post de servire și clientul care este nou sosit înaintea stației întrerupe serviciul clientului aflat acolo, pentru a începe propriul său serviciu. Clientul înlăturat este repus în prima poziție a firului de așteptare și el își va relua serviciul din punctul de întrerupere atunci cînd clientul ce l-a întrerupt și-a terminat serviciul său. Și aici distribuția timpilor de serviciu este dată de o lege de tip Cox.

Stațiile de tipul 2 permit folosirea terminalelor utilizate pentru aplicații în timp partajat sau a altor ansamble de unități în care fiecare proces (în sens informatic) poate obține servicii fără să aștepte. Stația de tip 3 este o idealizare a unei unități centrale, în care alocarea se face prin cuante; idealizarea se referă la faptul că durata cuantei este infinitezimală, neintervenind nici un „overhead" pentru comutarea clienților la epuizarea cuantei. Limitările stațiilor de tipul 1 constituie motivația principală a folosirii unor metode de aproximare (difuzie, iterative) la găsirea soluției stării de echilibru a rețelei.

3.1.7. METODE DE APROXIMARE

Singurele rețele de șiruri de așteptare cărora li se cunoaște soluția sub forma explicită sînt rețelele lui Jackson și BCMP. Dacă se dorește să se studieze rețele mari cu cîteva sute de șiruri de așteptare calculul soluțiilor poate deveni extrem de complicat sau chiar imposibil, chiar și pentru calculator. În cazul că interesează doar disciplina FIFO, rețelele cu mai mult de trei șiruri cărora li se cunoaște soluția au caracteristicile următoare: distribuțiile timpilor de serviciu sînt date de legi exponențial-negative; există doar o clasă de clienți în sistem (cînd se află mai multe, timpii de serviciu sînt independenți de clase); sosirile sînt poissoniene; șirurile de așteptare au toate capacitatea infinită. După cum se observă, în această situație numărul cazurilor cărora li se cunoaște rezolvarea este foarte redus. Mai mult, soluțiile cunoscute explicit sînt constituite în probabilități condiționate, care nu sînt necesare în practică. De aceea, s-au căutat metode aproximative de aflare a soluțiilor. O clasificare neexhaustivă a metodelor întrebuințate pentru modelarea matematică a sistemelor de șiruri de așteptare este dată în figura 3.4. Procedura de tratare a oricărei probleme ce comportă șiruri de așteptare cuprinde următoarele faze: a) analiza sistemului real și obținerea modelului (o rețea de șiruri de așteptare); b) scrierea ecuațiilor care descriu modelul (obținerea modelului matematic). Parametrii ce caracterizează modelul (legi de servire, fluxuri de intrare) sînt obținuți prin măsurători sau prin cunoașterea modelului real; c) rezolvarea sistemului de ecuații. În faza b trebuie să se definească mai întîi — plecînd de la model — ansamblul stărilor sistemului, apoi să se calculeze relațiile care leagă între ele probabilitățile acestor stări, ceea ce este posibil doar dacă modelul este markovian. În caz contrar se poate recurge la unul din următoarele demersuri: să se definească sublanțuri Markov in-

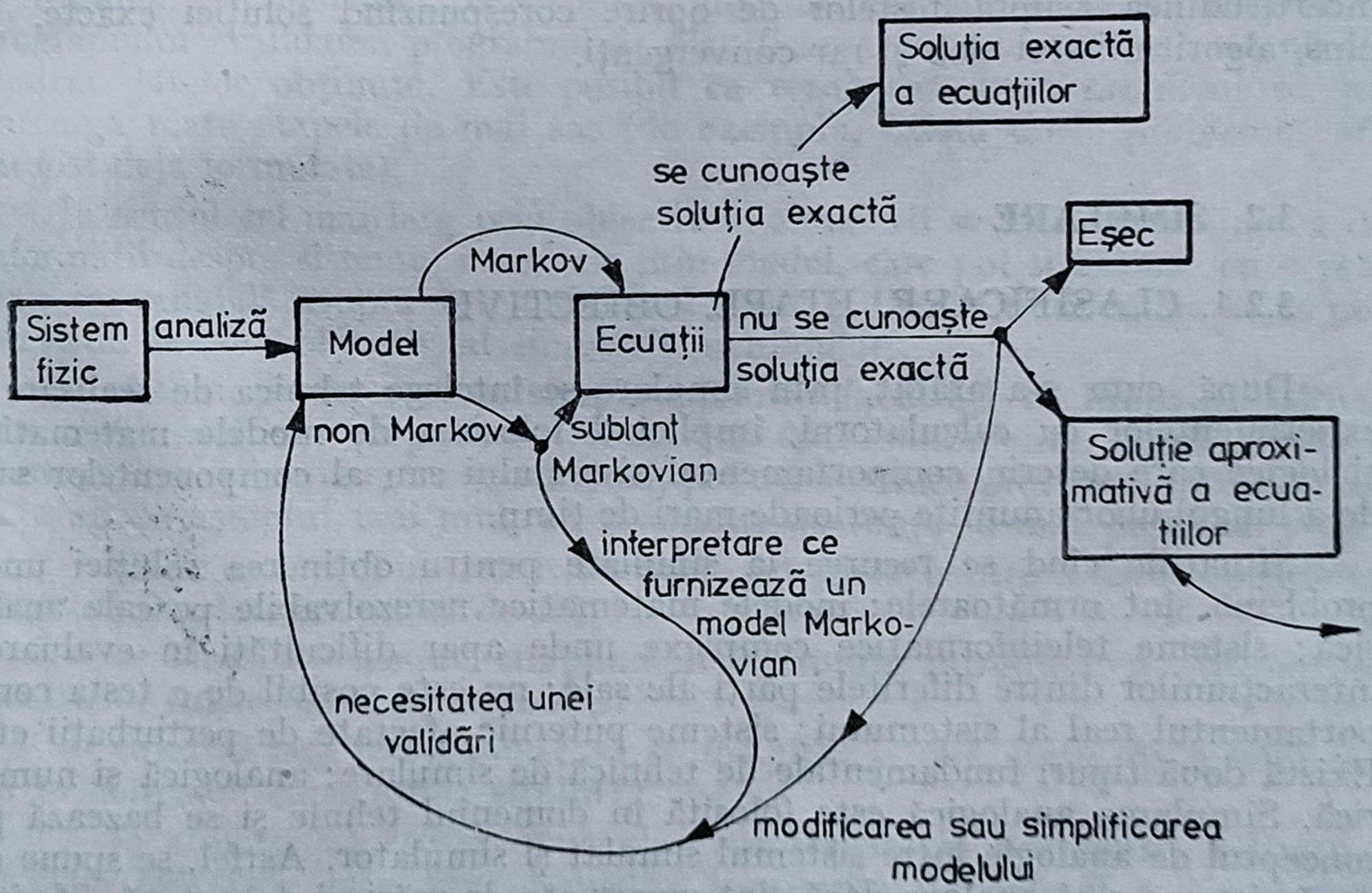


Fig. 3.4. Clasificarea metodelor de analiză a sistemelor cu șiruri de așteptare

cluse; să se ajungă la un model markovian, creînd mai multe stări fictive ale sistemului (de exemplu, prin descompunerea legii Erlang-n, în n servicii exponențiale în tandem); să se simplifice modelul presupunînd, de exemplu, că o stație asigură un serviciu exponențial distribuit, că un șir de așteptare este nelimitat și că un flux este Poissonian. În acest caz, precum și în cel anterior, trebuie să se încerce o evaluare a acestei soluții inexacte față de soluția exactă; să se modifice modelul pentru ca să devină calculabil, omițînd sau adăugînd anumite elemente.

Dintre metodele aproximative utilizate în practică pot fi amintite:

- *Metoda descompunerii*, care constă în studierea fiecărui șir din rețea, unul cîte unul, cu caracterizarea fluxurilor intermediare. Metoda se aplică doar rețelelor deschise.

- *Metoda valorilor medii* are ca scop unic studierea valorilor medii, timpul de răspuns și numărul de clienți într-un șir. Metoda se aplică atît la rețelele deschise, cît și la cele închise.

- *Metoda agregării* recurge la studiul sistemului pe grupe de stații determinate prin interacțiunile foarte slabe cu exteriorul. Se aplică rețelelor de șiruri de așteptare markoviene.

- *Metoda izolării* procedează la izolarea fiecărui șir și la studierea sa formală, ținînd seama de parametrii de interacțiune cu exteriorul. Rezolvarea explicită este realizată apoi pe ansamblul sistemului.

- *Metoda iterativă* constă în găsirea pentru fiecare stație a unei suite de rate a sosirilor, astfel încît comportamentul stației să fie din ce în ce mai apropiat de comportamentul real. Decizia apropierii comportamentului calculat față de cel real (necunoscut) se face prin îndeplinirea anumitor condiții (problema testului de ieșire). Dificultatea acestor metode iterative constă în incertitudinea asupra testelor de oprire corespunzînd soluției exacte. În plus, algoritmii sînt slab și rar convergenți.

3.2. **SIMULARE**

3.2.1. CLASIFICARE. ETAPE. OBIECTIVE

După cum s-a arătat, prin simulare se înțelege tehnica de realizare a experimentelor cu calculatorul, implicînd folosirea de modele matematice și logice care descriu comportamentul sistemului sau al componentelor sale de-a lungul unor anumite perioade mari de timp.

Situațiile cînd se recurge la simulare pentru obținerea soluției unor probleme sînt următoarele: modele matematice nerezolvabile pe cale analitică; sisteme teleinformatice complexe unde apar dificultăți în evaluarea interacțiunilor dintre diferitele părți ale sale; nu este posibil de a testa comportamentul real al sistemului; sisteme puternic afectate de perturbații etc. Există două tipuri fundamentale de tehnică de simulare: analogică și numerică. Simularea analogică este folosită în domeniul tehnic și se bazează pe conceptul de analogie între sistemul simulat și simulator. Astfel, se spune că două sisteme sînt analoge dacă sînt guvernate de mărimi de natură diferite, dar comportarea lor este descrisă de aceeași structură matematică. Simularea

numerică cuprinde totalitatea procedeelor matematice destinate studiului comportării în timp a sistemelor reale cu ajutorul calculatoarelor electronice numerice, presupunîndu-se că în evoluția acestor sisteme intervin și elemente aleatoare. Sînt cunoscute două variante a simulării numerice: simularea de tip joc și analiza Monte-Carlo.

Simularea de tip joc (operațional gaming) este specifică acelor situații ce se caracterizează printr-un conflict între doi sau mai mulți parteneri ori între om (care trebuie să ia unele decizii) și natură (ce oferă omului mai multe variante, din care el trebuie să o aleagă pe cea mai convenabilă). Metoda se folosește pe larg într-o serie de aplicații în domeniul militar și în economie pentru problemele de organizare și conducere a activității economice în întreprinderi sau ramuri industriale.

Analiza Monte-Carlo este o tehnică de simulare legată mai mult de probleme cu caracter aleator. Metoda asociază problemei respective un model aleator, și — prin generarea unor variabile aleatoare ale cărei momente, densități sau funcții de repartiție sînt legate funcțional de soluție — se realizează experiențe și se furnizează informații asupra soluției problemei deterministe.

Simularea de tip Monte-Carlo se distinge prin simplitate, generalitate și dependență slabă dintre timpul de calcul și dimensiunea problemelor. Dezavantajul metodei constă în slaba convergență către soluție și — de aici — un timp calculator relativ ridicat.

După T. H. Naylor realizarea unui experiment de simulare se desfășoară în mai multe etape: formularea problemei; colectarea și prelucrarea primară a datelor; formularea modelului de simulare; estimarea parametrilor, formularea ecuațiilor și ipotezelor statistice; evaluarea performanțelor modelului și testarea parametrilor; conceperea algoritmului simulării și codificarea programului; validarea programului; planificarea experiențelor de simulare; analiza datelor obținute. Este posibil ca rezolvarea unei probleme să nu parcurgă toate etapele de mai sus (de exemplu, există unele probleme care au fost deja formulate).

În sensul cel mai larg prin obiectiv al simulării se înțelege obținerea de informații despre sistemul construit prin model, care pot fi extinse cu o precizie convenabilă asupra sistemului real studiat. Dintre problemele ce se pot constitui într-un obiectiv al simulării menționăm:

— cîte stații de servire simple (numite facilități în limbajul GPSS) sau stații multiple (depozite) trebuie instalate suplimentar dacă numărul tranzacțiilor (clienților) nu se schimbă, pentru a reduce procentul tranzacțiilor care au de așteptat mai mult de n unități de timp, la mai puțin de y%?;

— cît de mult trebuie mărit un fir de așteptare pentru a reduce pierderea tranzacțiilor la mai puțin de x%?;

— ce efect are asupra timpului de așteptare introducerea priorităților pentru anumite clase de tranzacții?;

— ce ar fi mai convenabil? să se instaleze o altă stație de servire sau să se mărească capacitatea șirului?;

— în situația că există două clase de tranzacții, o anumită calitate a servirii (măsurată prin timpul mediu de așteptare) poate fi realizată mai ieftin cînd fiecare clasă este înzestrată cu cîte un grup separat de servire sau dacă se introduce un singur grup care să servească ambele clase?

3.2.2. INSTRUMENTE DE SIMULARE

Pentru scrierea mai rapidă a programelor de studiere a modelelor de simulare pe calculatoare electronice există o gamă largă de instrumente specifice numite limbaje de simulare. Acestea sînt limbaje specializate care facilitează legătura dintre utilizator și calculator și minimizează volumul de muncă necesar scrierii și punerii la punct a programelor. Uneori durata de execuție a programelor scrise în unele limbaje de simulare este mare față de cea a programelor codificate în limbaje universale (PASCAL, FORTRAN, PL/1). Cu toate acestea utilizarea limbajelor spacializate este în final mult mai economică, întrucît punerea la punct a programelor este relativ simplă. Există diferite criterii de clasificare a limbajelor de simulare. Astfel, în funcție de tipul sistemelor studiate limbajele pot fi continue sau discrete, deterministe ori stohastice, complet sau vag definite ; dacă se ia în considerare natura reprezentării fenomenelor din sistem se disting limbaje de simulare orientate pe tranzacții (cereri de servire) — caz în care momentele cînd se produc evenimentele (schimbarea stării sistemului) sînt considerate caracteristici ale unor entități a sistemului — și limbaje orientate pe reprezentarea activităților — cînd se urmărește executarea diverselor activități de către cantități diferite, înregistrîndu-se succesiunea lor. În funcție de reprezentarea în timp a evenimentelor în cadrul modelului se evidențiază limbaje în cadrul cărora ceasul de simulare este independent de eveniment și altele la care evenimentele sînt marcate de tactul ceasului simulării (exemplul limbajului GPSS). După numărul de faze există limbaje cu două sau trei faze, ce pot fi în continuare grupate în cele orientate pe activități și specializate pe evenimente.

La limbajele de simulare cu două faze prima determină actualizarea ceasului de simulare, iar în a doua se caută evenimentul care trebuie să se producă în momentul următor. Limbajele cu trei faze necesită un timp de calcul relativ mic și asigură o reprezentare facilă a datelor. Prima fază constă în actualizarea ceasului simulării, cea de a doua în identificarea momentelor apariției evenimentelor, iar a treia, în căutarea evenimentelor ce trebuie să se producă în continuarea momentelor determinate în faza a doua.

Există două mari categorii de limbaje de simulare: convenționale și orientate pe blocuri.

La limbajele convenționale (FORTRAN, PL/1, PASCAL, ALGOL) programatorul trebuie să rezolve probleme specifice legate de tehnica simulării (comanda ceasului simulării, prelucrarea statistică a datelor, tehnica simulării etc.). Au o mare flexibilitate în efectuarea unor calcule de gestiune economică sau științifică, dar volumul activității de analiză — programare implicate este mare. În schimb scrierea programelor de simulare în limbaje orientate pe blocuri nu are acest dezavantaj datorită existenței unor instrucțiuni sau blocuri ce codifică anumite funcții. Există limbaje orientate pe sisteme discrete (GPSS, SIMSCRIPT, SIMULA, GASP, SIMUB, CSL) și pe sisteme continue (CSMP, LSSC, ANALOG).

3.2.3. SIMULAREA SISTEMELOR DISCONTINUE. LIMBAJUL GPSS

Sistemele discontinue (discrete) își modifică — prin salturi — starea la anumite intervale de timp, eventualele stări tranzitorii care apar nefiind luate în considerație.

Modelele ce descriu sisteme discrete se construiesc cu ajutorul unor elemente specifice numite resurse, clienți, cereri, stații, cozi etc. Implementarea în practică a acestor modele se face cu ajutorul unor limbaje de programare ca FORTRAN, SIMULA, SIMSCRIPT, GPSS etc. Limbajul GPSS (General Purpose Simulation System) este folosit pentru studiul fenomenelor ce apar în sistemele de servire, unde — în funcție de factorul de servire — pot apărea cozi (șiruri) de așteptare. Există mai multe variante dialectale, printre care amintim GPSS V — implementat pe sisteme IBM 360/370, SIAS (SIEMENS) și SIMUB (Felix-C 256/1024).

Sistemele ce se simulează în GPSS sînt descrise ca diagrame bloc, unde blocurile reprezintă activitățile, iar liniile ce le unesc desemnează secvențele în care decurg evenimentele. Întrucît limbajul este fundamentat pe metoda descriptivă, fiecărui bloc îi este asociat un nume ce descrie acțiunea respectivă, blocul fiind reprezentat printr-un simbol grafic. Deplasarea prin sistem este realizată de abstracțiuni elementare numite ENTITATI, care în simulare poartă numele de TRANZACTII și care depind de natura sistemului de simulat (de exemplu, înregistrările într-un sistem de calcul, mesajele într-un sistem de comunicații).

Secvența de evenimente în timp real este reflectată prin mișcarea tranzacțiilor dintr-un bloc în altul în timp simulat. Limbajul are 14 tipuri de entități, fiecare din ele fiind caracterizată de atribute (proprietăți) ce îi definesc starea la un moment dat. Aceste atribute pot fi adresate extern prin mnemonice și pot avea valori numerice sau logice.

Programul GPSS menține o înregistrare a timpilor la care trebuie să se producă evenimentele, ce le execută apoi în secvența cerută. Se simulează un „ceas" care înregistrează momentul care a fost atins în modelul sistemului real. Valoarea acestui ceas este denumită „ceasul absolut al simulării". Mai există și un alt ceas, „relativ", care poate fi apelat de programator. El este un atribut numeric standard și se identifică cu simbolul Cl.

În programul GPSS nu se simulează sistemul la intervale constante, egale cu unitatea de timp aleasă, ci se avansează ceasul absolut la momentul în care trebuie să se producă evenimentul următor, obținîndu-se astfel un sistem de simulare cu ceas variabil. Pe măsură ce timpul (simulat) trece, entitățile interacționează, ceea ce produce transformări — numite evenimente — în valorile numerico-logice asociate. Entitățile se pot grupa în lanțuri ori seturi, a căror componență este posibil să se modifice în timp. Cele 14 tipuri de entități GPSS sînt: de bază (blocurile și tranzacțiile); de echipament (facilități, depozite și comutatori logici); de calcul (variabile aritmetice, variabile booleene, funcții); statistice (cozi și tabele de frecvență); de referință (valori memorate și matrice memorate); tip-lanț (lanțuri de utilizator și grupuri).

Programul scris în GPSS este format dintr-un șir de enunțuri — prin care se specifică și se descriu blocurile lanțului de prelucrări ale tranzacțiilor conform diagramei-bloc — și din enunțuri de control. Toate lanțurile GPSS se înscriu pe formulare de programare între coloanele 1 și 72, ținîndu-se seama de următoarele reguli:

— coloana 1 este rezervată specificării cartelelor-comentariu;

— coloanele 2—6 conțin etichetele asociate blocurilor. Etichetele sînt opționale, fiind formate din minimum 3 caractere literate și sînt atribuite de utilizator blocurilor apelate în program;

— coloana 7 este rezervată;

— din coloana 8 se scrie numele blocului, compus din maximum 5 caractere, din care primele 3 sînt obligatoriu litere;

— coloanele 19—71 sînt rezervate operanzilor descriptori ai blocurilor. Separatorul este virgula; între operanzi nu se permite blancul. Dacă lipseşte un cîmp descriptor se trec două virgule alăturate. Primul blanc întîlnit constituie o indicaţie a terminării descrierii blocului.

Ansamblul asociază automat fiecărui bloc un număr de identificare. Mişcarea prin sistem se face de la blocul n la blocul $n + 1$ (în afara situaţiei salturilor (ne)condiţionate). După ce a fost decodificat programul, blocurile sînt listate în ordine crescătoare a acestor numere. O diagramă-bloc poate conţine pînă la 1 000 blocuri. Unitatea adresabilă este byte-ul, instrucţiunile fiind pe cuvînt (4B), semicuvînt (2B) şi pe byte. Programul începe cu enunţul SIMULATE şi se încheie cu enunţul END.

3.3. MĂSURAREA ŞI ANALIZA REZULTATELOR

3.3.1. STATISTICI FURNIZATE DE LIMBAJELE DE SIMULARE

Limbajele de simulare folosite frecvent pentru analiza dinamicii activităţilor şi comportărilor STI dau utilizatorului o serie de rapoarte statistice standard şi opţionale privind unele măsuri obţinute ca rezultat al simulării. În continuare vor fi exemplificate aceste situaţii statistice pe cazul limbajului GPSS, utilizat în simularea proceselor descrise de modele cu şiruri de aşteptare.

- **Rapoarte statistice standard**

— *blocurile*, în care se indică numărul tranzacţiilor care au trecut prin fiecare bloc şi numărul tranzacţiilor ce se află în fiecare bloc în momentul tipăririi raportului;

— *facilităţile* ilustrează numărul tranzacţiilor care au ocupat facilitatea, gradul ei de ocupare şi timpul mediu petrecut de tranzacţii în facilitate;

— *depozitele* prezintă capacitatea acestora, numărul de tranzacţii care au trecut prin fiecare depozit, timpul mediu petrecut de tranzacţii în depozit, gradul mediu de utilizare al acestor resurse şi numărul de tranzacţii din fiecare depozit la momentul publicării raportului;

— *comutatorii logici*, care se află în starea „poziţionat" (setat);

— *cozile*, unde se dau informaţii privind lungimea maximă a cozii, lungimea actuală, timpul mediu petrecut în coadă, numărul de tranzacţii cu aşteptare nulă în coadă, procentajul acestora faţă de totalul tranzacţiilor care au trecut prin blocul de ataşare la coadă şi timpul mediu de aşteptare efectivă a tranzacţiilor în coadă.

- **Rapoarte statistice opţionale**

În simulare tabelele de distribuţie se folosesc pentru a obţine distribuţia de frecvenţă a unui anumit argument, ce poate fi orice atribut numeric stan-

dard. Variabilele care au frecvența cea mai mare de utilizare a tabelelor GPSS/360 sînt: timpul mediu de tranzit prin întregul sistem sau pe segmente din sistem (M1); distribuția lungimii cozilor (QA_i) și timpii de așteptare (QT_i, QX_i); distribuția conținutului mediu al depozitelor (SA_i); distribuția intervalului de timp între intrările într-un tabel (sau a intervalului de timp între sosirile în blocul TABULATE). Aceste rapoarte se obțin cu ajutorul blocului TABULATE și al cartelei de definiție TABLE. De exemplu, secvența:

```
          TABULATE    1
          _________
          _________
          _________

1         TABLE       M1,   0,   30,   20
          _________
```

arată că se va obține în tabelul 1 distribuția frecvențelor timpului de tranzit al tranzacțiilor pînă la blocul TABULATE, de la începutul simulării, din 30 în 30 unități de timp, pentru 20 de intervale.

3.3.2. INTERPRETAREA REZULTATELOR SIMULĂRII

În simularea evenimentelor discrete se utilizează un model al sistemului care simulează evoluția în timp a principalilor parametri caracteristici ai sistemului sub influența variabilelor exogene (sau variabilele de intrare) și a interacțiunilor între variabilele interne ale sistemului. Simularea poate fi dirijată prin evenimente sau prin ceas de simulare a scurgerii timpului. În primul caz starea sistemului este examinată numai în momentele cînd au loc modificări în variabilele sistemului, metoda fiind curent folosită în simularea STI. În cazul al doilea timpul este avansat de un ceas de simulare, starea sistemului fiind examinată în fiecare unitate de timp, pentru determinarea tranzițiilor de stare care pot avea loc. În această abordare se consumă mult mai mult timp de calcul al calculatorului față de primul caz.

Ca în orice experiență statistică datele obținute prin simulări nu pot fi interpretate decît drept niște estimări ale proceselor sau variabilelor dorite. În continuare vor fi prezentate cîteva elemente privitoare la calculul intervalului de încredere asociat unei variabile estimate prin măsurători într-un simulator și o metodă relativ simplă de a-l obține.

Fie X lungimea unui fir de așteptare la un moment dat t în cursul unei simulări. Pentru a estima valoarea sa medie $M(X)$ se efectuează n simulări independente, cu diferite valori ale variabilelor pseudoaleatoare. Se obține o serie de valori ce pot fi considerate ca o realizare a unui șir de variabile aleatoare independente, $(X_1, X_2, ..., X_n)$, de aceeași lege ca și X. O estimare a lui $M(X)$ este dată de estimatorul

$$\overline{X} = \frac{1}{n} \sum_{i=1}^{n} X_i \qquad (3.24)$$

Variabilei aleatoare $\overline{X}$ (numită medie empirică) i se asociază dispersia empirică:

$$S = \frac{1}{n} \sum_{i=1}^{n} (X_i - \overline{X})^2. \tag{3.25}$$

Datorată independenței v.a. X_i din cele două relații de sus se obține:

$$M(X) = M(\overline{X}) \tag{3.26}$$

$$D^2(X) = nD^2(\overline{X}) = \frac{n}{n-1} D^2(S) \cdot \tag{3.27}$$

Folosind inegalitatea lui Cebîșev se poate scrie:

$$P(|\overline{X} - D^2(X)| \geqslant b) \leqslant \frac{1}{n-1} \cdot \frac{D^2(S)}{b^2} \cdot \tag{3.28}$$

Dacă dispersia $D^2(X)$ este finită, termenul al doilea din inegalitate tinde la zero cînd n tinde la infinit.

Se afirmă că pentru un interval de încredere de $\pm b$, coeficientul de încredere în procente este $1 - \frac{1}{n-1} \cdot \frac{D^2(S)}{b^2} \cdot$ Pentru obținerea unui interval de încredere mai fin, se observă că distribuția lui $\overline{X}$ tinde către o lege normală, întrucît este vorba de un șir de v.a. independente și distribuite identic. Se poate scrie:

$$P\left\{\frac{\overline{X} - D^2(X)}{\sqrt{S/(n-1)}} < d\right\} = \frac{1}{\sqrt{2\pi}} \int_{-d}^{d} e^{-\frac{y^2}{2}} dy = A \cdot \tag{3.29}$$

Întregul din membrul drept se numește nivel de încredere. Valoarea sa A reprezintă probabilitatea ca $D^2(X)$ să se găsească în intervalul aleator:

$$[\overline{X} - d\sqrt{S/(n-1)}, \quad \overline{X} + d\sqrt{S/(n-1)}], \tag{3.30}$$

numit interval de încredere.

Alegerea nivelului de încredere A (sub formă de procent) permite determinarea lui d prin folosirea tabelelor legii normale. Valoarea luată de dispersia empirică S este utilizată împreună cu d pentru a estima lărgimea intervalului de încredere $2 \cdot d\sqrt{S/(n-1)}$. Intervalul este obținut din (3.30), folosind aceste rezultate și valorile măsurate ale lui $\overline{X}$ și S. Astfel de exemplu presupunînd un nivel de încredere de 95%, inegalitatea lui Cebîșev dă un nivel de încredere de $9\sqrt{S/(n-1)}$, în timp ce utilizînd legea normală se obține o lărgime de aproximativ $4\sqrt{S/(n-1)}$. Diferența rezultă din faptul că folosirea intervalului (3.30) este justificată doar pentru n foarte mari. Concluzia este că noțiunile de coeficient și interval de încredere servesc la măsurarea exactității unei estimări.

3.3.3. MONITOARE DE MĂSURĂ

După realizarea sa practică, performanțele unui STI ce au fost estimate încă din faza proiectului logic și modificate apoi pe parcursul testărilor primare sau prin simulare pot fi acum măsurate efectiv în contextul unei funcțio-

nări reale. Există o întreagă tehnologie de măsurare a mărimilor parametrilor caracteristici unui STI care se bazează — în fapt — pe tehnicile generale de măsurare a activității S.O., cunoscută fiind interacțiunea strînsă între activitatea STI și procesele ce se derulează în interiorul S.O. Se disting: tehnici de măsurare programată ce folosesc monitoare software pentru măsurarea activității STI și tehnici de măsurare instrumentală care utilizează monitoare hardware la măsurarea performanțelor STI.

Tehnicile de măsurare programată. Există două concepte în această activitate de măsurare: *interceptarea evenimentelor cheie* ce apar în sistem și *eșantionarea stărilor* corespunzătoare activității sistemului, ambele concepte fiind implementate sub forma unor monitoare software încorporate în SO.

Tehnica de interceptare (permanentă) folosește module sursă scrise în limbaj de asamblare ce sînt apoi înserate în punctele cheie ale nucleului rezident al SO. Aceste module ce se constituie într-un monitor extractor, preiau controlul după apariția unor evenimente semnificative din activitatea sistemului, înregistrînd informațiile pe un suport extern (fig. 3.5). Datele sînt prelucrate de modulul monitor analizor, ce asigură analistului rezultate pe baza cărora se iau decizii despre comportarea sistemului. Monitorul extractor ia în considerație evenimente precum: gestiunea activității sistemului de I/E; gestiunea activității task-urilor active în sistem; gestiunea alocării resurselor de memorare; gestiunea orologiilor interne; frecvența utilizării legăturilor; frecvența de ocupare a zonelor de memoria-tampon; frecvența incidentelor de orice natură (retransmisia și reinițializarea pe legături, comutarea pe echipamente de rezervă etc.).

Precizia extractorului este dată de curbele de încredere trasate în funcție de timpul de extragere.

Tehnica de eșantionare permite ca la intervale de timp prefixate de utilizator sau la inițiative SO controlul să fie trecut modulelor de extragere, care culeg din tabele date referitoare la sistemul de calcul gazdă, la legătura de date

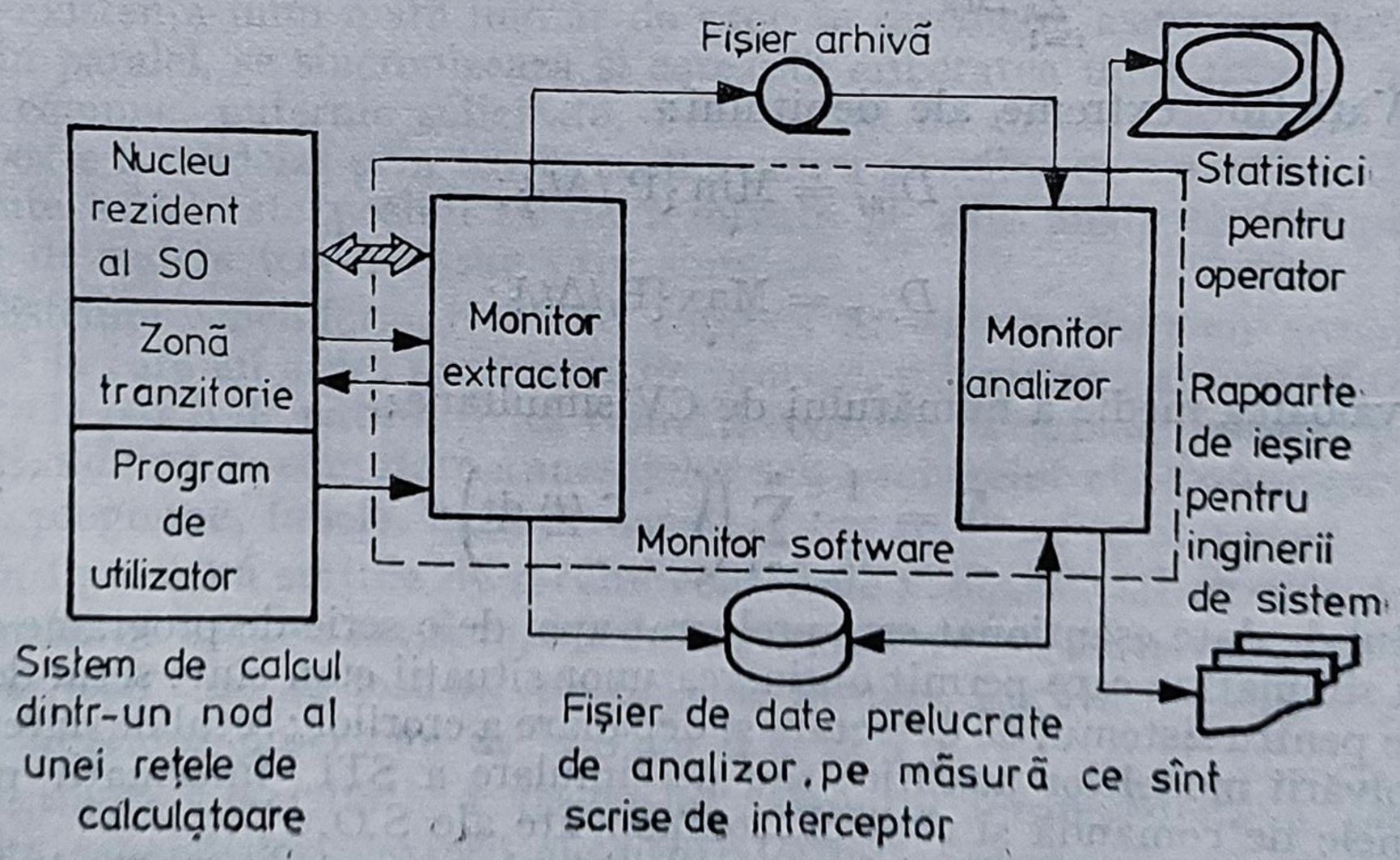


Fig. 3.5. Monitor software de interceptare permanentă pentru măsurarea performanțelor STI

etc. și le depun pe o memorie externă. Pentru a nu se pierde prea multe date utile trebuie ca intervalul de eșantioane să fie suficient de mic. Datele eșantionate sînt: starea UC; starea șirurilor de operații de I/E; starea șirurilor de erori la operațiile de I/E; starea operațională a sistemului de I/E local; gradul de ocupare a memoriilor tampon de noduri; nivelul traficului pe linii; numărul de retransmisii a unor pachete afectate de erori; starea nodurilor și a liniilor.

Să considerăm, de exemplu, o rețea cu comutare de pachete și să analizăm ceva mai pe larg care sînt mărimile de performanță ce pot fi obținute. Pe durata unor intervale de timp disjuncte Δt_i $(i = 1, ..., n)$ pentru fiecare nod din rețea se pot măsura următorii indicatori caracteristici:

— numărul N_i al noilor circuite virtuale (CV) stabilite pe durate Δt_i;

— numărul P_i de pachete care traversează nodul în Δt_i;

— intervalul de timp τ_{ij} de stabilire a fiecărui CV realizat la cererea unui utilizator conectat direct la acel nod, în timpul intervalului Δt_i;

— numărul C_i (t) de CV care traversează nodul la momentul t al intervalului Δt_i.

Cu acești indicatori caracteristici se pot calcula

— timpul mediu $\overline{\tau}$ de stabilire a unui CV:

$$\overline{\tau} = \frac{\sum_{i=1}^{n} \sum_{j=1}^{n} \tau_{ij}}{\sum_{i=1}^{n} N_i} \tag{3.31}$$

— debitul mediu efectiv al nodului;

$$\overline{D} = \frac{\sum_{i=1}^{n} P_i}{\sum_{i=1}^{n} \Delta t_i} < \text{pachete/unitatea de timp} > \tag{3.32}$$

— Variațiile extreme ale debitului:

$$D_{inf} = \operatorname{Min}_i \{P_i/\Delta t_i\}; \tag{3.33}$$

$$D_{sup} = \operatorname{Max}_i \{P_i/\Delta t_i\}; \tag{3.34}$$

— valoarea medie a numărului de CV simultane:

$$\overline{N} = \frac{1}{n} \cdot \sum_{i=1}^{n} \left(\int_{\Delta t_i} C_i(t)\, dt \right). \tag{3.35}$$

Setul de date eșantionat este prelucrat apoi de o serie de programe analitice sau simulative care permit obținerea unor situații cum sînt: setul de evenimente pentru sistemul de detectare și depanare a erorilor; setul de date necesar rezolvării modelelor analitice sau de simulare a STI, informații pentru programele de comandă și control, componente ale S.O.

Analizînd cele două tehnici expuse mai înainte se observă că tehnica de eșantionare reduce — față de tehnica de interceptare — cantitatea de timp

pentru obținerea unor informații cu grad mediu de detaliere, elimină necesitatea de a avea puncte de interceptie și nu necesită adrese precise de la care să fie eșantionate datele. Dezavantajul tehnicii de eșantionare este că timpul de execuție al rutinei de măsurare este mare, crescînd proporțional cu rata de eșantionare.

Tehnicile de măsurare instrumentale sînt implementare prin monitoare hardware, care constituie componente ale arhitecturii interne a sistemelor de calcul dintr-un STI sau echipamente portabile, adaptabile la diverse standarde și topologii de rețele. Ele operează asupra evenimentelor la nivel de semnal, putînd fi monitoare de tip rezumativ sau de tip dinamic. Cele de tip rezumativ furnizează informații la modul „cît de des" sau „cît de mult" are loc un anumit eveniment în activitatea rețelei fără a preciza „cînd" s-a petrecut evenimentul respectiv. Monitoarele de tip dinamic colectează și furnizează date asupra momentelor „cînd" s-au produs evenimente importante și le înmagazinează în zone-tampon proprii, fiind prelucrate ulterior. În practică se folosesc de obicei monitoare hardware mixte, se combină funcțiile celor două tipuri menționate și care sînt folosite pentru: măsurători asupra rețelei de transmisie a datelor și a procesoarelor frontale cuplate la canalele de I/E; măsurarea numărului de circuite virtuale asigurate de o legătură logică în cadrul protocolului HDLC; echilibrarea optimă a sarcinilor în STI etc.

3.4. APLICAȚII

3.4.1. CLASE DE MODELE PENTRU SISTEME DE COMUNICAȚIE CONȚINUTE ÎN MODELUL DE REFERINȚĂ OSI

O trăsătură caracteristică pentru un sistem teleinformatic (STI) o constituie existența unui mare număr de procese distincte, asincrone, care evoluează în paralel, se sincronizează și așteaptă eliberarea unor resurse de prelucrare comune, puternic solicitate. Fenomenul de dispută ce apare în acest caz, poate fi modelat prin teoria cozilor sau a șirurilor de așteptare. Modelele obținute astfel este posibil să fie rezolvate pe cale analitică (cel puțin din punct de vedere teoretic) sau prin simulare.

Sistemul teleinformatic este compus în esență dintr-un ansamblu de resurse la care au acces un număr de procese, mesaje sau activități (fig. 3.6) Resursele pot fi de natură fizică (unități centrale, terminale, unități periferice de I/E, noduri de comutare a mesajelor sau pachetelor etc.) sau logice (informații, programe, tabele, baze de date).

În figura 3.6 stațiile de servire constituie resursele, iar șirurile de așteptare înaintea fiecărei stații reprezintă activitățile care au nevoie de serviciul stațiilor, sau care au fost servite deja de resursă. Legăturile între stații ilustrează mișcarea proceselor de la o sursă la alta.

Extrema diversitate a resurselor, evoluția imprevizibilă a proceselor datorită solicitărilor aleatorii ale utilizatorilor și marea complexitate a STI au condus la formalizarea comportamentului lor prin metode stochastice.

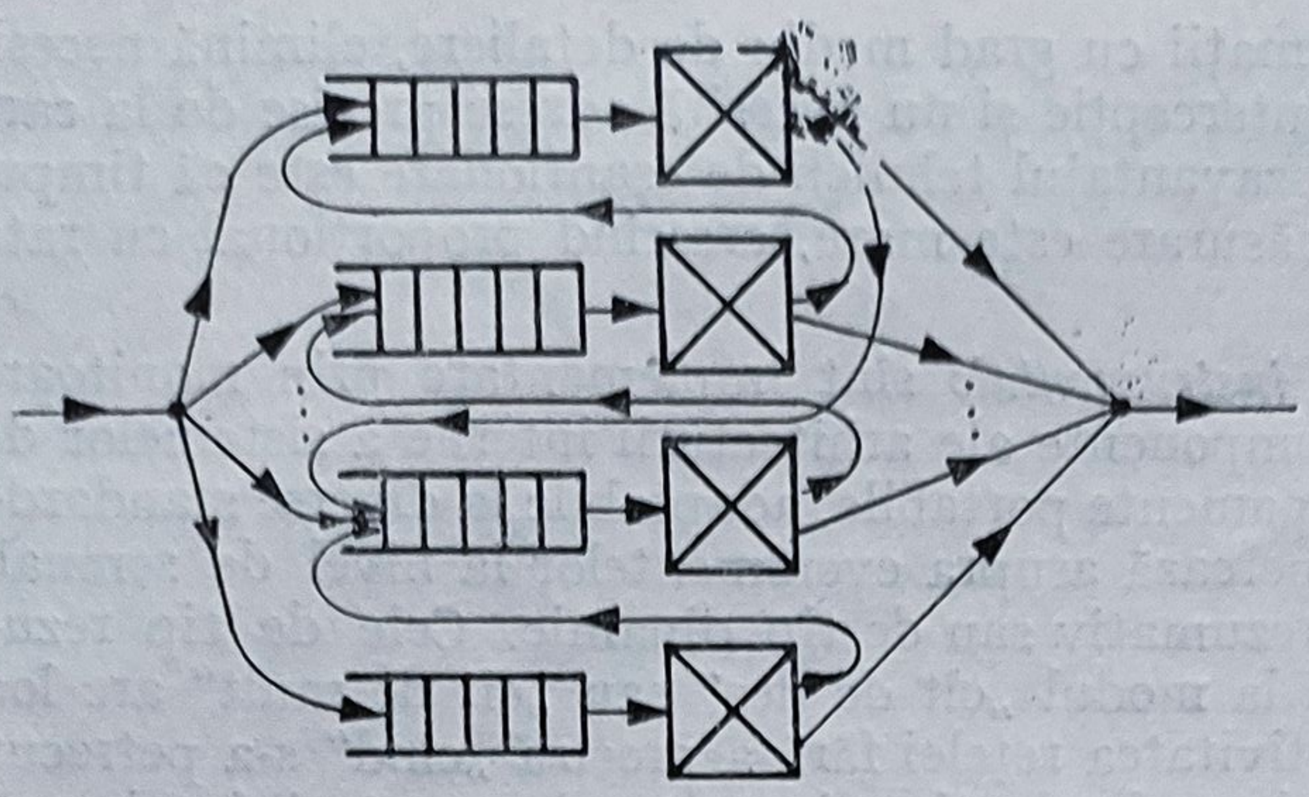


Fig. 3.6. Reprezentarea unui STI sub forma unei rețele de șiruri de așteptare

Studiul acestor sisteme se realizează pe baza rețelelor de șiruri de așteptare. O rețea de șiruri de așteptare este alcătuită dintr-un ansamblu finit de stații de servire și un ansamblu infinit de clienți. Ea este caracterizată de un proces stohastic reprezentînd sosirile clienților în rețea, timpii de servire aleatoare a clienților de către stații, de procesul stohastic al dirijării clienților de la o stație la alta și de disciplinele de servire efectuate de stații. Există rețele închise — ce cuprind nu număr finit de clienți care circulă în rețea — și rețele deschise, implicînd sosiri și plecări ale clienților în/din rețea.

• Modele de activități în Modelul de referință ISO-OSI

Încă din anii '60, cînd au fost concepute primele sisteme de teleprelucrare a informației, s-a încercat ca pe baza folosirii unor instrumente precum teoria șirurilor de așteptare și simularea aleatoare a evenimentelor discrete, să fie evaluați diferiții indicatori de performanță a acestor sisteme (timpul de răspuns, timpul de traversare a rețelei, gradul de utilizare a sistemului pentru o încărcare dată, disponibilitatea sistemului etc.). Pentru analiza și evaluarea STI se folosesc diferite metode de modelare precum rețele de șiruri de așteptare markoviene, metode de aproximație ca metoda valorilor medii, metoda agregării, metoda izolării, metode iterative, metode exacte (P—2).

Dacă se consideră activitățile specifice nivelelor Modelului de Referință ISO, se apreciază că au fost realizate și rezolvate parțial sau complet modele doar pentru nivelele inferioare ale acestui standard arhitectural. Problematica modelării activităților nivelelor 5—7 constituie o problemă centrală a cercetării actuale din domeniu.

Pentru nivelul 2 — Legătura de date — modelele utilizate sînt de tipul „lanțuri Markov", în timp ce în nivelul 3 — Rețea — se ajunge la modele cu rețele deschise de șiruri de așteptare la care, pentru rezolvare, se folosește presupunerea independenței mesajelor. În cadrul nivelului 4 — Transport — apar rețele multiple de șiruri de așteptare grupate în tandem, accentul cercetării fiind pus pe reducerea complexității algoritmilor de rezolvare a modelelor. Printre protocoalele acestui nivel un loc important îl reprezintă cele care se ocupă de controlul ferestrei fluxului de pachete și administrarea memoriilor tampon (buffere) din nodurile de comunicație a RC.

3.4.2. MODELAREA ȘI EVALUAREA FACTORULUI DE ANTICIPARE ÎNTR-UN PROTOCOL HDLC

Prin factor de anticipare în protocolul HDLC se înțelege numărul maxim de pachete care pot fi transmise dintr-un nod fără ca vreunul din ele să fi fost achitat. La protocolul HDLC valoarea normalizată pentru acest factor (fereastra protocolului) este 7, existînd și o valoare mai mare egală cu 127.

Fie n factorul de anticipare. În scopul definirii stării emițătorului și a receptorului este necesar să se considere două niveluri distincte: cel logic — corespunzînd fenomenelor în care intervine UC din nod — și nivelul fizic — aparținînd fenomenelor legate în principal de cuploarele de linie. Se face presupunerea că aceste două niveluri sînt independente. Stările utilizate de procedura HDLC sînt ilustrate în figura 3.7. Timpii t_{01} și t_{11} sînt timpii de gestiune ai emițătorului și ai receptorului, iar t_{00} și t_{10} reprezintă timpii de gestiune necesari descifrajului, controlului etc., la sosirea într-un nod. Se consideră că nu există erori pe linie. Timpul de transmitere a unui pachet are valoarea:

$$S = \frac{1}{\mu} + \frac{1}{\nu},$$

unde $\frac{1}{\mu}$ este timpul de transmisie efectiv pe linie de către emițător, iar $\frac{1}{\nu}$ constituie ansamblul celorlalți timpi de așteptare, întîrzierea de propagare, timpii de gestiune etc. Paralelismul este efectiv la nivelul acestor timpi $\frac{1}{\nu}$, astfel încît modelul protocolului HDLC poate fi reprezentat cu ajutorul unor stații de servire în paralel (fig. 3.8).

Modelul este format dintr-o primă stație compusă dintr-un șir de așteptare cu o singură stație de servire și un al doilea element alcătuit din n stații de scriere în paralel fără șir de așteptare. Atunci cînd un client își termină serviciul său μ și găsește plină a doua stație își reîncepe serviciul și continuă astfel pînă cînd va depista un loc în ea. Fie p probabilitatea ca n clienți să

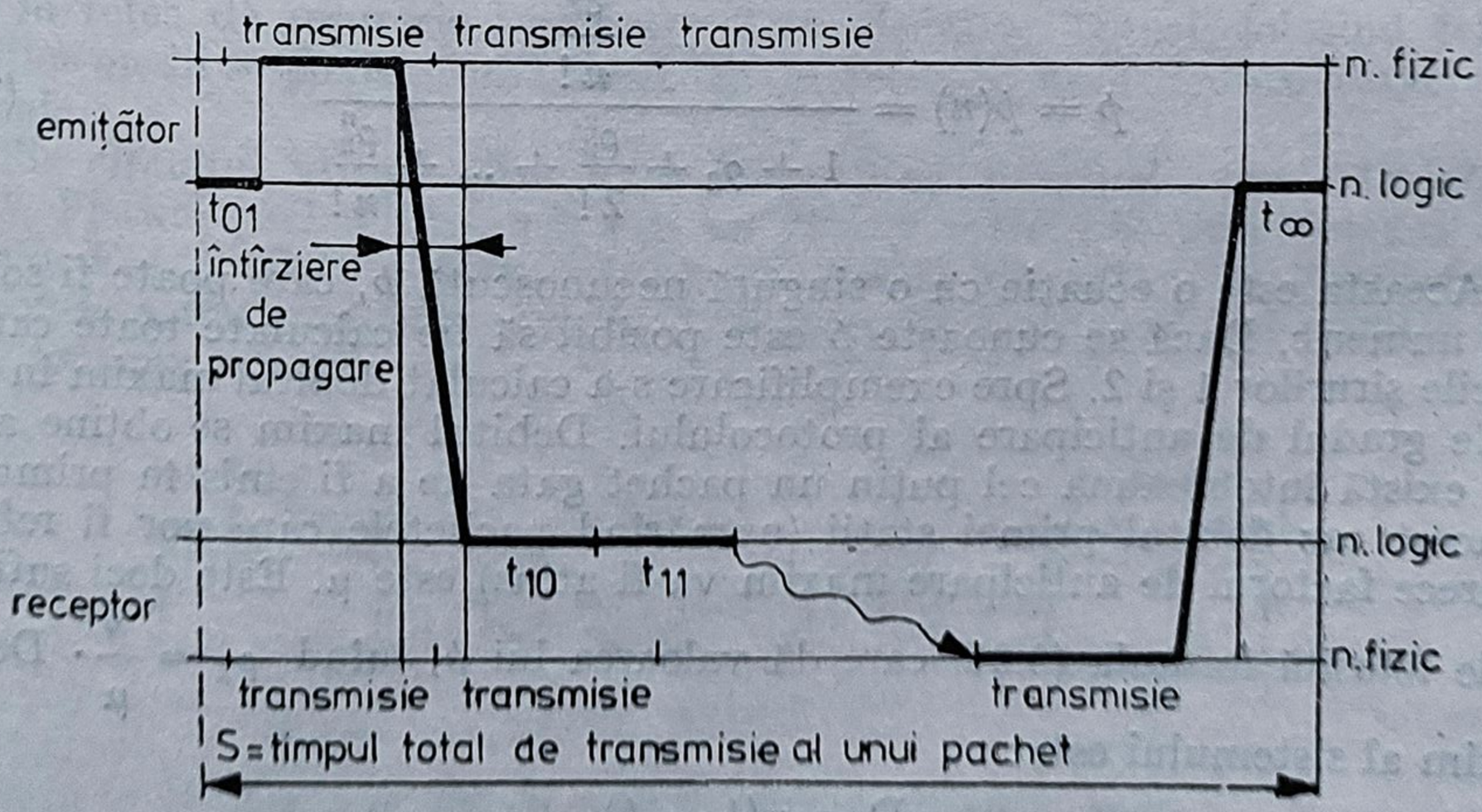


Fig. 3.7. Stările folosite de procedura HDLC

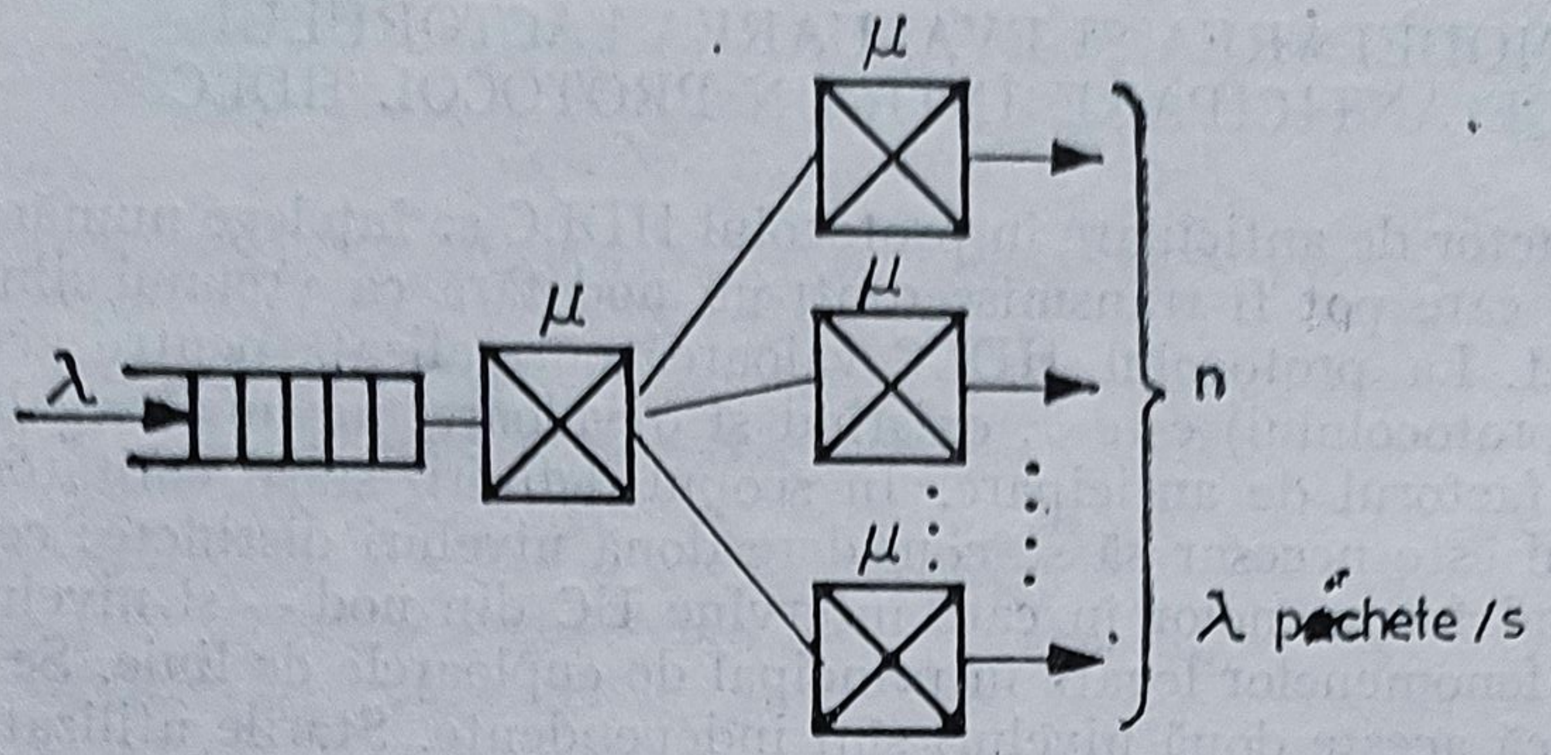


Fig. 3.8. Modelul protocolului HDLC

fie în a doua stație. Atunci timpul de servire echivalent care cuprinde ansamblul timpilor petrecuți de un client în stația de servire 1 și 2 are valoarea medie $\frac{1}{\mu(1-p)}$.

Interfața de intrare poate fi modelată prin ansamblul clienților ce se prezintă pentru a obține un loc în șir. Un același client este numărat de atîtea ori de cîte ori intră în șir. Rata sosirilor este în acest caz $\lambda/(1-p)$.

Studiul primului și al celui de al doilea șir poate fi făcut separat, însă în funcție de parametrul p. O soluție explicită se obține rezolvînd simultan ecuațiile generate de cele 2 subsisteme de servire. Primul șir este de tip (M|M|1): (FIFO|∞|∞|):

$$\tilde{\lambda} = \frac{\lambda}{1-p}; \quad \tilde{\mu} = \mu(1-p); \quad \rho_1 = \frac{1}{\tilde{\mu}}; \quad \rho_2 = \frac{\tilde{\lambda}}{\nu}.$$

Al doilea fir de așteptare este de tipul (M|G|C): (C|O|n) pentru care se știe că:

$$p = p(n) = \frac{\frac{\rho^2}{n!}}{1 + \rho_2 + \frac{\rho_2^2}{2!} + \ldots + \frac{\rho_2^n}{n!}} \tag{3.36}$$

Aceasta este o ecuație cu o singură necunoscută p, care poate fi soluționată numeric. Dacă se cunoaște p este posibil să fie calculate toate caracteristicile șirurilor 1 și 2. Spre exemplificare s-a calculat debitul maxim în funcție de gradul de anticipare al protocolului. Debitul maxim se obține atunci cînd există întotdeauna cel puțin un pachet gata de a fi emis în primul șir. În acest caz debitul primei stații (numărînd pachetele care vor fi refuzate deoarece factorul de anticipare maxim va fi atins) este μ. Este deci suficient să fie folosită formula (3.36) care dă valoarea lui p, luînd $\rho_2 = \frac{\nu}{\mu}$. Debitul maxim al sistemului este:

$$D = \mu(1-p).$$

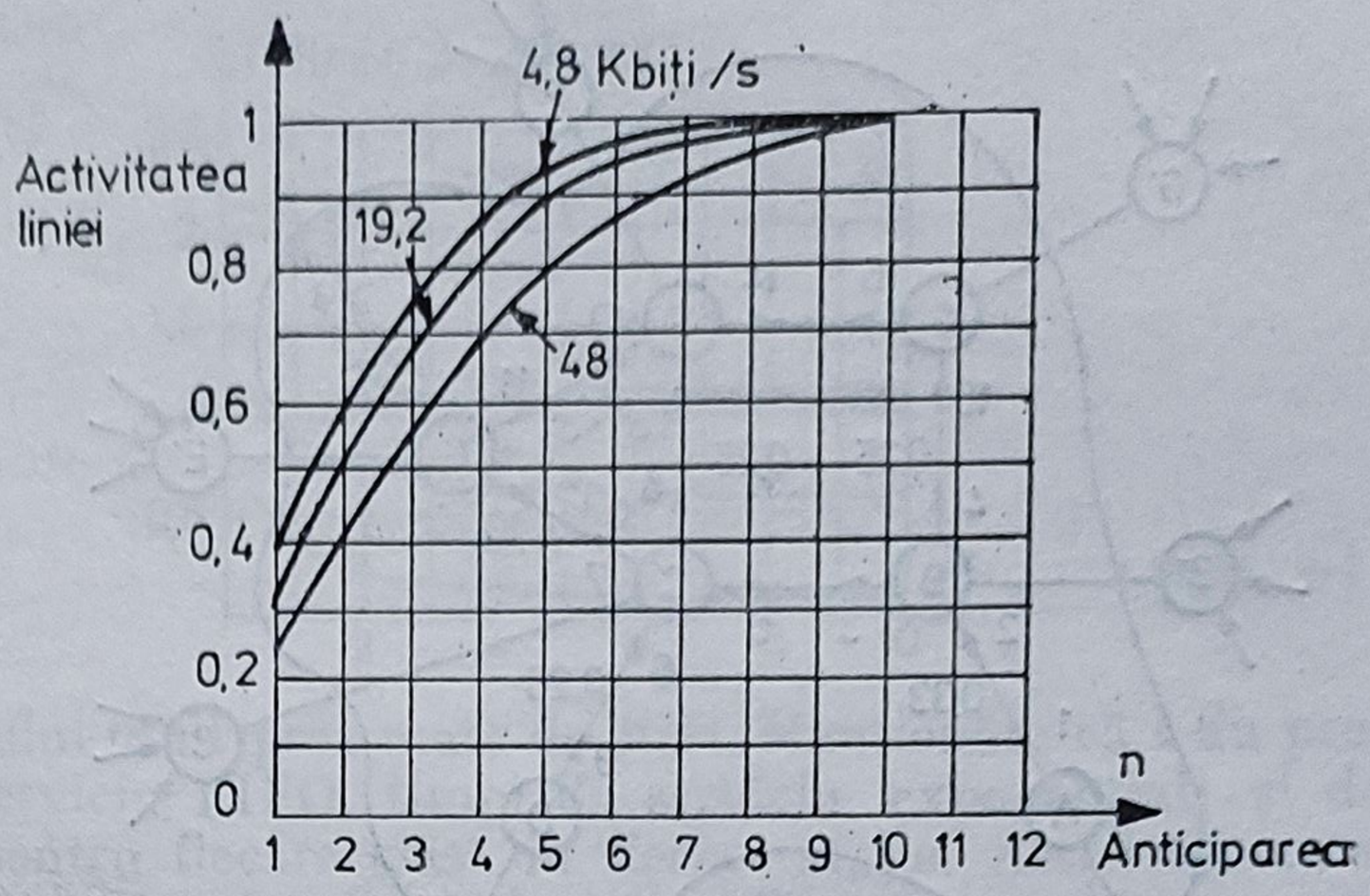


Fig. 3.9. Dependența între activitatea liniei, dată de debitul normalizat și gradul de anticipare al protocolului HDLC

În figura 3.9. este trasată dependența între activitatea liniei $(1 - p)$ și gradul de anticipare al protocolului HDLC pentru trei tipuri distincte de linii: 4,8 kb/s, 19,2 kb/s și 48 kb/s. S-a presupus că lungimea medie a unui pachet este de 1 000 biți și că distribuția emisiei sale este exponențială.

3.4.3. STUDIUL PERFORMANȚELOR UNEI REȚELE DE CALCULATOARE CU CIRCUITE VIRTUALE

Se știe că un pachet nu poate intra într-o rețea de calculatoare decît dacă i-a fost afectat un circuit virtual. La începutul lucrului este nevoie în general de un *pachet de apel* care să marcheze circuitul virtual (el va fi format doar dacă resursele sistemului sînt suficiente) și de un *pachet de terminare* ce șterge circuitul virtual (CV). Circuitul virtual este traseul din rețea pe care îl vor parcurge pachetele ce urmează să-i fie afectate. El mobilizează un anumit număr de resurse în rețea, cum sînt de exemplu memoriile — tampon din nodurile de comunicație.

Studiul rețelelor cu CV este dificil de realizat datorită dependenței intrărilor în rețea de resursele interioare deja afectate. Totuși, folosind tehnica valorilor medii se poate studia acest tip de rețea și apoi deduce caracteristicile sale.

Fie circuitul virtual (fig. 3.10) între două terminale sau calculatoare-gazdă. Fiecare pachet transmis într-un circuit virtual se numerotează ciclic modulo $F + 1$ ($F \in Z_+^*$). Este permis un factor de anticipare (fereastră) F, putîndu-se astfel transmite pînă la F pachete, fără ca să fi fost primite achitările corespunzătoare.

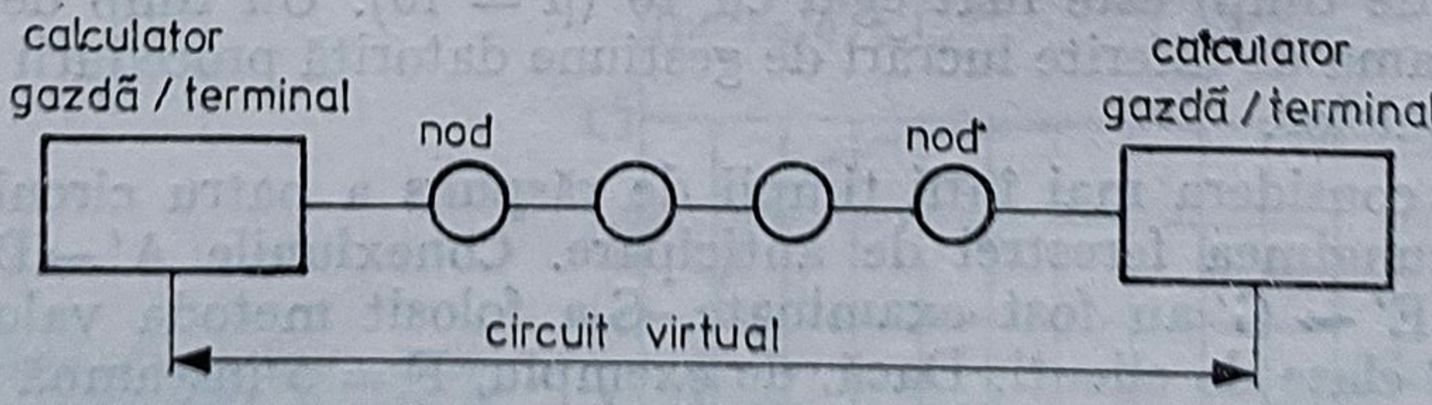


Fig. 3.10. Circuit virtual între două stații gazdă

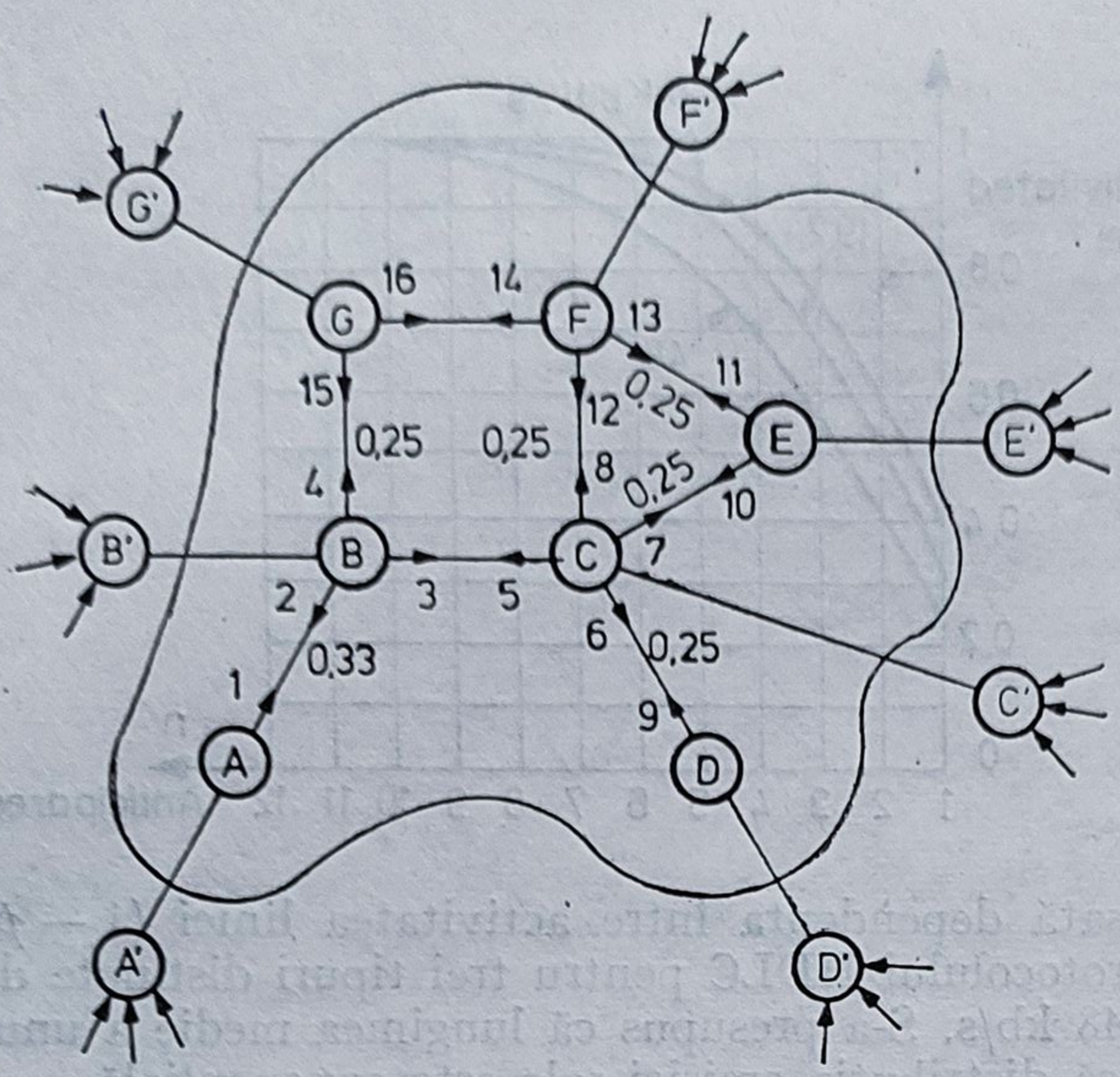


Fig. 3.11. Rețea de calculatoare cu circuite virtuale

TABELUL 3.1. Dirijări complementare în rețeaua de circuite virtuale din fig. 3.11

Sursă	Destinație	Via
A'	F'	C
B'	F'	C
C'	G'	F
D'.	G'	F
E'	C', D'	B
F'	A', B'	C

Pentru a controla debitul în această rețea trebuie schimbate valorile factorului de anticipare și numărul de canale virtuale.

Se va considera rețeaua din figura 3.11, unde nodurile A-G sînt reprezentate de minicalculatoare care memorează pachete.

Nodurile A'—G' reprezintă echipamente terminale. Se presupune că dirijarea pachetelor este fixă și este dată în tabelul 3.1. Dacă o rută nu este indicată în această tabelă, se alege itinerarul cel mai scurt. Vom considera că lungimile pachetelor sînt exponențial distribuite și că aceste pachete se prezintă echipamentelor terminale, urmînd un proces Poisson. De asemenea, pachetele ce sosesc din exterior au o probabilitate egală de a merge către unul din celelalte 6 calculatoare-gazdă.

Rețeaua din figura 3.11 poate să fie modelată de un sistem cu 16 fire de așteptare interne și de 7 fire de intrare (cîte unul pentru fiecare calculator gazdă). Se presupune că nu există fire de ieșire „nod-gazdă", întrucît ele corespund unor linii mult mai rapide, iar cele 42 de rute posibile „gazdă-gazdă", posedă fiecare un circuit virtual, cu același factor de anticipare. Calea virtuală ce posedă o fereastră de anticipare F poate fi înlocuită prin F circuite virtuale cu o fereastră egală cu 1. În exemplul dat, toate liniile au aceeași capacitate de transfer; n (numărul mediu de pachete care pot fi transmise în unitatea de timp) este luat egal cu 10 ($\mu = 10$). Un timp de serviciu de 0,1 s ține seama de diferite lucrări de gestiune datorită procedurii de transmisie și controalelor.

Se vor considera mai întîi timpii de răspuns a patru circuite virtuale, schimbînd lungimea ferestrei de anticipare. Conexiunile A' → D', A' → F', D' → G' și E' → G' au fost examinate. S-a folosit metoda valorilor medii. Există 42 F clase de clienți. Dacă, de exemplu, F = 5 înseamnă că ar fi 210 ($42 \cdot 5 = 210$) clase de clienți, situație în care nu se poate aplica teorema

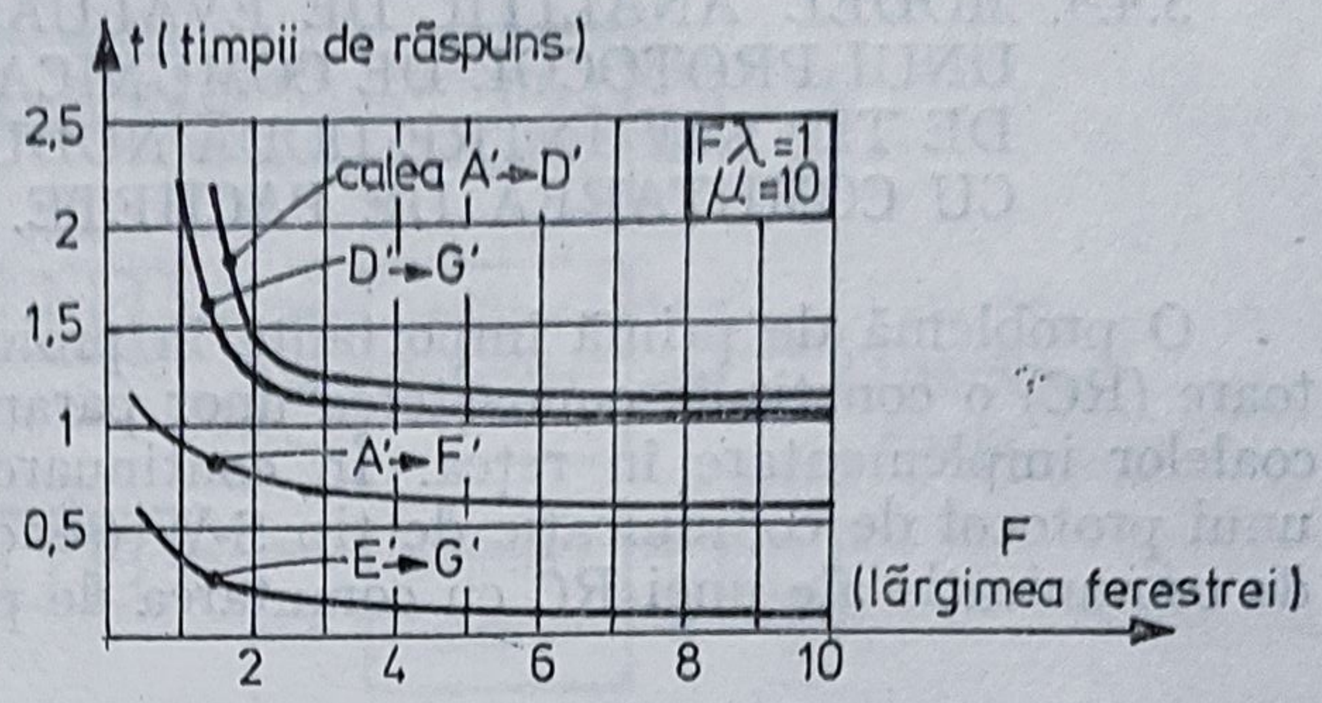


Fig. 3.12. Ansamblu de soluții pentru timpul de răspuns în funcție de lărgimea ferestrei

BCMP din cauza numărului mult prea mare de stări. Menționăm că s-au presupus o disciplină de serviciu FIFO, timpi de serviciu exponențiali și de aceeași valoare medie pentru fiecare clasă în fiecare stație. Astfel teorema BCMP ar fi din punct de vedere conceptual, aplicabilă.

În exemplul de mai sus coeficientul de trafic este $\rho_{ir} = 1$ dacă clasa clientului r trece prin firul i. Pentru ca încărcarea în rețea să fie constantă vom considera că $\lambda F = 1$, unde λ reprezintă rata sosirilor clienților din clasa r. Se cunoaște că timpul mediu de răspuns al unei stații i dintr-o rețea generală este suma timpilor medii de serviciu și a timpilor medii de așteptare înainte de a fi servit, adică:

$$M(T_i^r) = M(S_i^r) \cdot \left(1 + \lambda \cdot \sum_{i=1}^{R} M(T_i^r)\right), \quad r = 1, \ldots, 42\,\mathrm{F}.$$

Se obține un sistem de 42 F ecuații cu tot atîtea necunoscute, ansamblul de soluții fiind dat în figura 3.12. Este evident că timpii de răspuns descresc neliniar atunci cînd anticiparea maximă se mărește. Astfel, de la o anticipare F = 4, timpii nu mai oscilează semnificativ.

Dependența dintre debitul maxim al fiecărui circuit virtual, în funcție de anticiparea maximă, este dată în figura 3.13. Debitul este limitat de circuitul virtual $A' \rightarrow D'$. Creșterea debitului în funcție de lărgimea ferestrei era previzibilă. Se remarcă și aici că pentru F ⩾ 4 debitul nu mai crește practic deloc, trăgîndu-se concluzia că nu se poate mări debitul acționînd doar asupra ferestrei de anticipare pe circuitul virtual respectiv. În plus, dacă se presupune că în acest exemplu un circuit virtual de fereastră F necesită rezervarea a F bufere în fiecare nod, atunci numărul de bufere va crește foarte mult. Astfel, pentru firul 3 de ieșire din nodul B există 8 CV, fiind deci necesare 8F bufere în vederea alocării tuturor circuitelor virtuale.

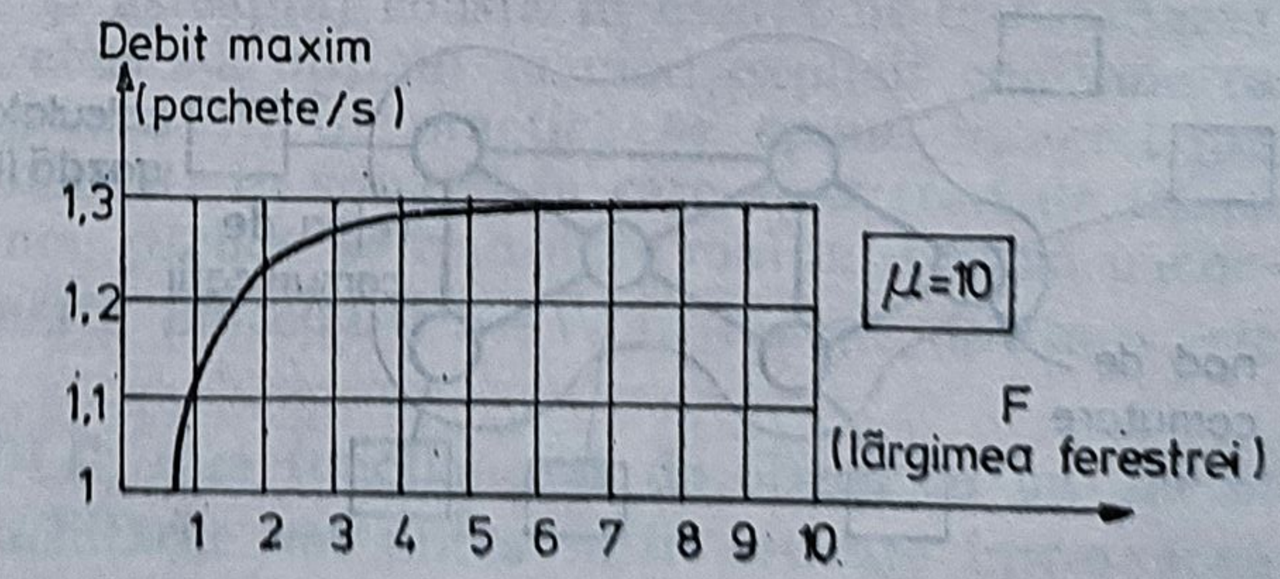


Fig. 3.13. Dependența între debitul maxim al fiecărui circuit și anticiparea protocolului

3.4.4. MODEL ANALITIC DE EVALUARE A PERFORMANȚELOR UNUI PROTOCOL DE COMUNICAȚIE DE TIP S-W ÎNTRE DOUĂ NODURI ALE UNEI RC CU COMUTAREA DE PACHETE.

O problemă de primă importanță în proiectarea unei rețele de calculatoare (RC) o constituie cunoașterea unor parametri de performanță a protocoalelor implementare în rețea. În continuare sînt evaluate performanțele unui protocol de comunicație de tip S-W (send and wait) între două noduri de comunicație ale unei RC cu comutarea de pachete [P—2].

• Modelarea funcționării unui nod din rețeaua de comunicație a RC

Într-o RC cu comutarea de pachete datele transmise sînt decupate în secvențe de biți de lungime fixă sau variabilă, numite pachete, a căror valoare maximă însă este o constantă a rețelei (de obicei, 256 biți). Pachetele sînt transportate în interiorul unor rețele de comunicații (fig. 3.14), ce realizează conectarea tuturor calculatoarelor din rețea prin intermediul unor noduri de comutare. Pachetele pleacă de la calculatorul-gazdă care le emite și traversează rețeaua de comunicație — din nod de comutare, în nod de comutare — pînă ajung la destinație. Un protocol de comunicație pentru această legătură de date trebuie să permită obținerea asigurării că pachetele care pleacă dintr-un nod ajung corect în nodul următor. În figura 3.15 este ilustrată organizarea logică principială a unui nod de comutație.

Funcțiile logice îndeplinite de un comutator sînt în esență următoarele:

• la fiecare linie de comunicații se asociază un emițător și un receptor. Emițătorul gestionează linia de ieșire și transmite pachetele pe linie, conform procedurii de comunicație folosite. Receptorul primește pachetele de pe linie și potrivit procedurii de comunicație utilizate, verifică corectitudinea pachetului, atribuie zonele-tampon libere și transferă pachetele către șirul de așteptare al comutatorului;

• funcții de comutație care există în gestiunea șirului unic de așteptare ale comutatorului, selectarea celei mai bune linii de ieșire către destinația pachetului, transferul pachetului către șirul de așteptare al acestei linii și distrugerea pachetului, dacă acest șir este plin;

• funcțiunile primare sînt constituite dintr-un ansamblu de servicii ce pot fi furnizate de nod. De exemplu, în rețeaua de comunicații CYGALE din

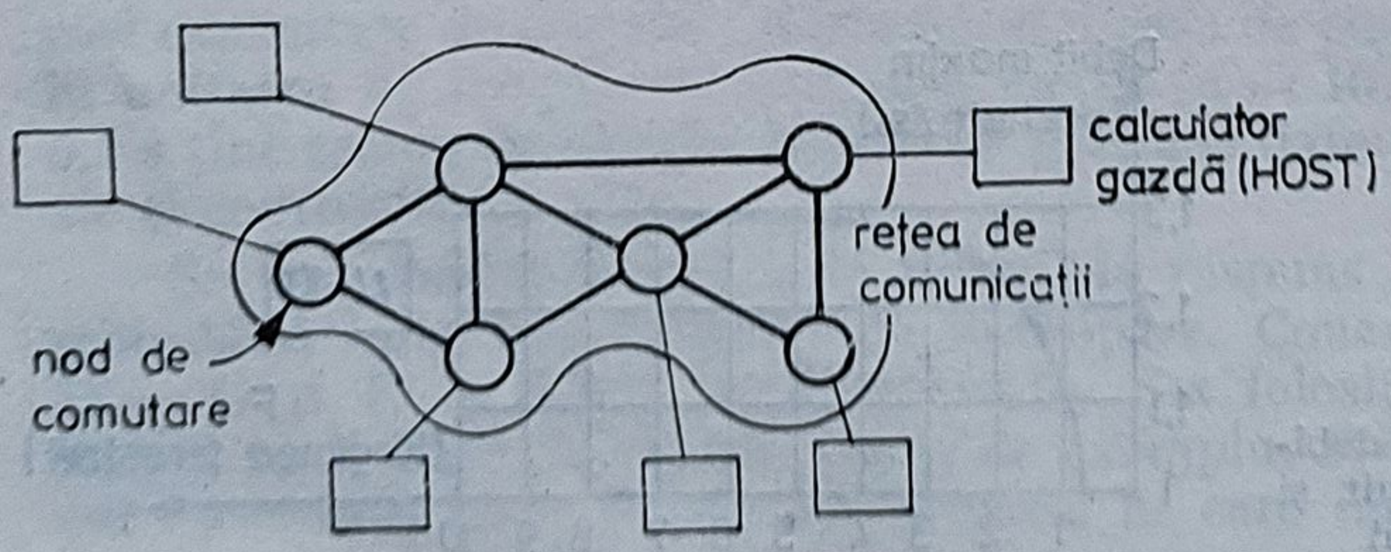


Fig. 3.14. Arhitectura unei rețele de calculatoare (RC) cu comutarea de pachete

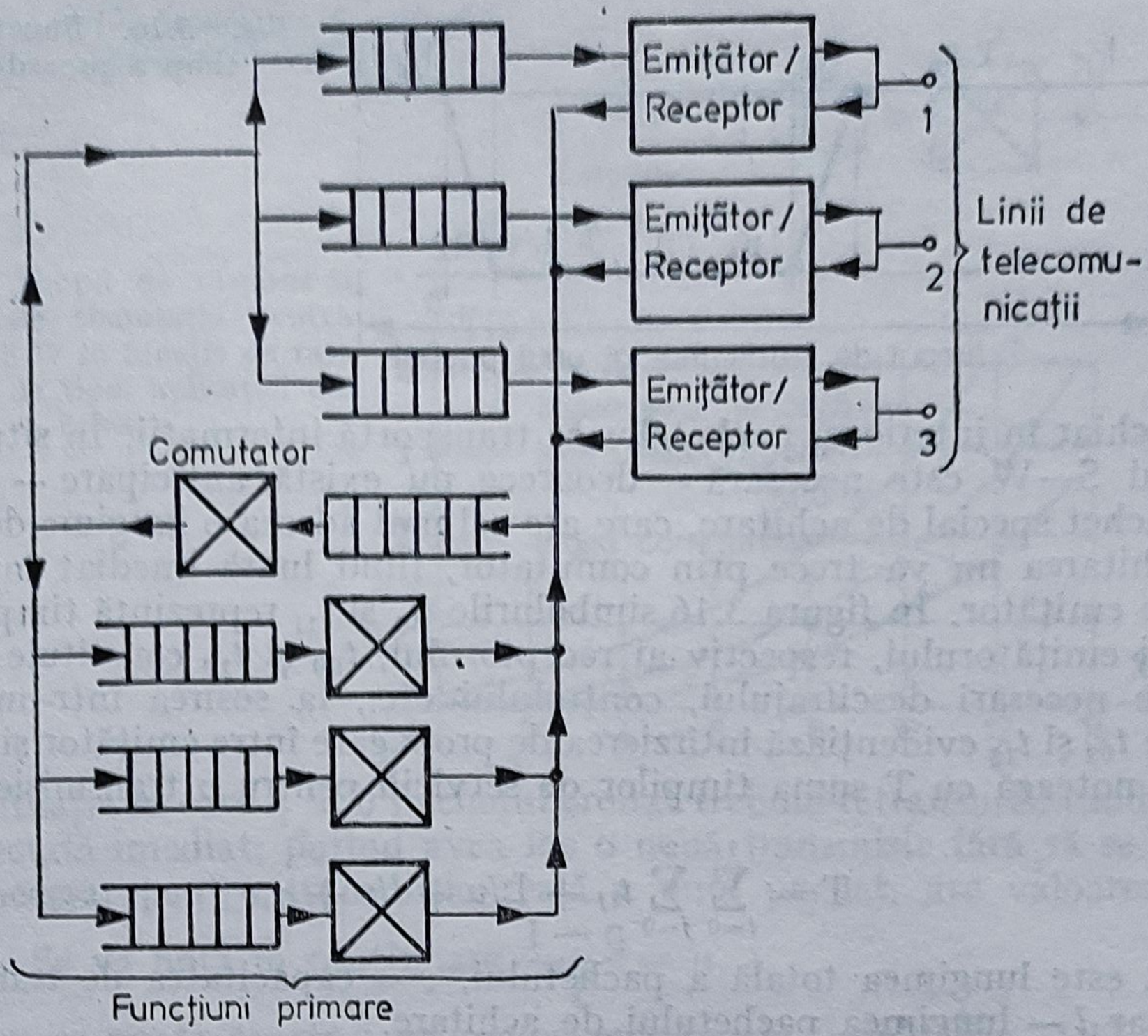


Fig. 3.15. Organizarea logică principială a unui nod de comutaţie

reţeaua de calculatoare CYCLADES, printre aceste funcţii primare pot fi amintite următoarele:

— STAT = trimite date statistice rezultate din exploatarea curentă a nodului;
— ROUTE — implementează un mecanism de dirijare adaptivă a pachetelor;
— HEURE = gestionează ora curentă;
— ECHO, = reţine la nodul de origine pachetele primite.

Funcţiile logice ale unui nod sînt afectate de priorităţi. Acestora li se asociază niveluri de întrerupere diferite, cărora le corespund timpi de gestiune sau timpi monitor (overhead).

Principalele priorităţi sînt în ordine descrescătoare, următoarele: (a) transmisia de către linii, (b) comutarea şi (c) funcţiunile primare.

Protocolul S—W (trimite şi aşteaptă) constă în esenţă în transmiterea unui nou pachet doar atunci cînd s-a obţinut în mod explicit confirmarea primirii corecte a pachetului precedent. În practică se spune în acest caz că pachetul precedent a fost achitat. În situaţia în care apar erori de transmisie înainte de a trimite un nou pachet, se transmite mai întîi copia pachetului eronat. Schema funcţionării procedurii S—W în funcţie de timp este prezentată în figura 3.16.

În cazul protocolului HDLC, care funcţionează de obicei cu un factor de anticipare egal cu şapte, achitările vor putea să fie inserate într-o zonă

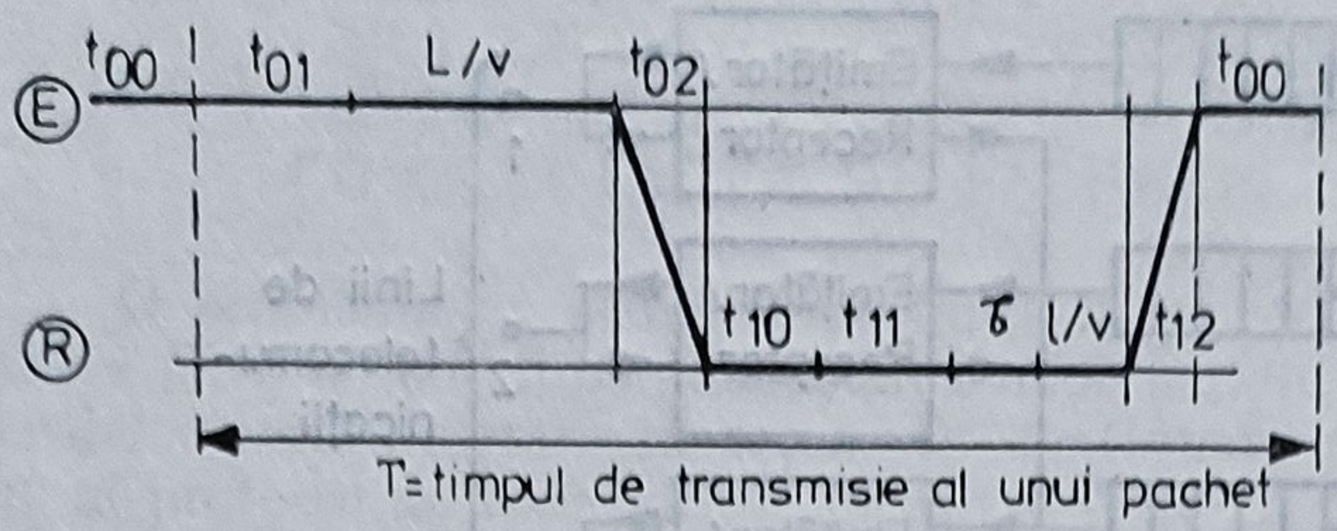


Fig. 3.16. Funcționarea în timp a procedurii S-W

situată chiar în interiorul pachetelor ce transportă informații. În situația protocolului S—W este necesară — deoarece nu există anticipare — utilizarea unui pachet special de achitare, care are cel mai adesea o lungime de 7 bytes.

Achitarea nu va trece prin comutator, fiind luată imediat în evidență de către emițător. În figura 3.16 simbolurile t_{01} și t_{11} reprezintă timpii de gestiune ai emițătorului, respectiv ai receptorului, t_{00} și t_{10} constituie timpi de gestiune necesari descifrajului, controlului etc., la sosirea într-un nod, în timp ce t_{02} și t_{12} evidențiază întîrzierea de propagare între emițător și receptor.

Se notează cu T suma timpilor de serviciu pentru o transmisie:

$$T = \sum_{i=0}^{1} \sum_{j=0}^{2} t_{ij} + L/v + l/v + \tau,$$

unde L este lungimea totală a pachetului, v — capacitatea de transmisie a liniei, iar l — lungimea pachetului de achitare.

Cantitatea τ constituie o întîrziere care are valoarea medie egală cu timpul de transmitere a unor jumătăți de pachet, dacă linia este foarte încărcată. Pentru a vedea ordinele de mărime a unora din acești timpi se vor considera rezultatele măsurătorilor efectuate asupra rețelei CYCLADES, deja amintită:

$$t_{00} = t_{10} \simeq 5 \text{ ms};$$

$$t_{01} = t_{11} \simeq 3 \text{ ms};$$

$$t_{02} = t_{12} = \begin{cases} 10 \text{ ms pentru linii cu debitul binar de 4,8 kbiți/s;} \\ 3 \text{ ms pentru alte linii.} \end{cases}$$

Timpii t_{02} și t_{12} iau în considerație timpii de gestiune ai modemurilor utilizate. Acești timpi de propagare corespund liniilor cu o lungime de aproximativ 500 km.

Dacă o eroare de transmisie are loc cu probabilitatea q, pachetul este distrus și este retransmisă o copie păstrată în nodul emițător.

Timpul de transmisie efectiv este $T = T(l, \tau)$, considerîndu-se în continuare că $l = 7$ bytes și că lungimea medie a unui pachet este de 100 biți. Valoarea timpului mediu de serviciu $\overline{T}$ în funcție de capacitatea liniei utilizate este dată în tabelul 3.2.

TABELUL 3.2. Valoarea timpului mediu de serviciu $\overline{T}$

Capacitatea liniei	4,8 kbiți/s	19,2 kbiți/s	48 kbiți/s
$\overline{T}$	360 ms	103 ms	54 ms

Fig. 3.17. Modelarea funcționării protocolului S-W

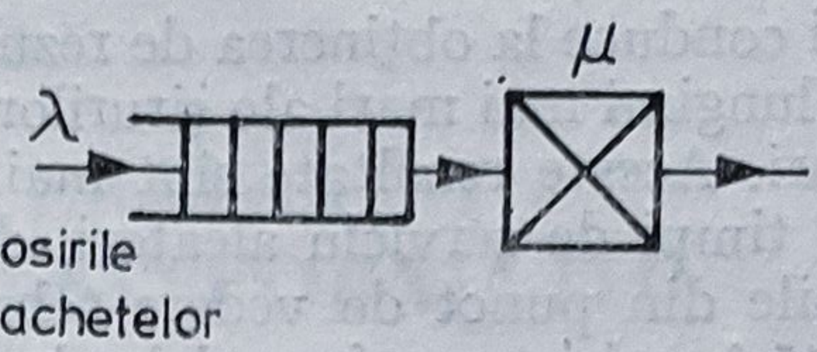
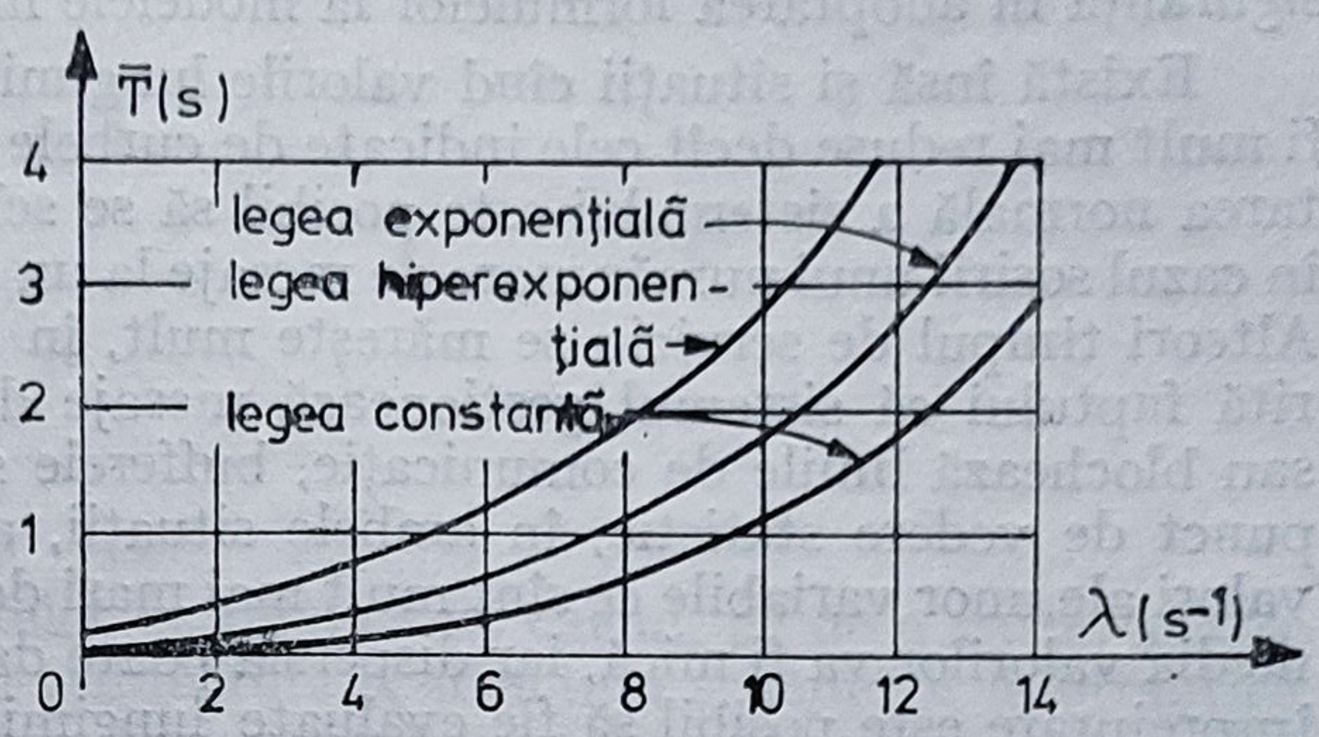


Fig. 3.18. Timpii de răspuns ai unui nod de comutație pentru procedura S-W în funcție de rata sosirilor și de tipul aplicației din rețea



În situația în care q = 0 pachetul eronat trebuie retransmis. Cînd eroarea este detectată imediat, putînd avea loc o nouă transmisie fără să se aștepte, timpul necesar pentru transmisia reală a unui pachet, are valoarea medie $\bar{T}/(1-q)$. Se va nota în continuare $\frac{1-q}{\bar{T}} = \mu$.

Acum se poate construi un model matematic simplu (fig. 3.17) care să reprezinte funcționarea protocolului S—W.

Frecvența de sosire a pachetelor în șirul de așteptare servite de o stație cu parametrul μ este λ. Vom presupune că sosirile pachetelor în nodul de comutație se efectuează conform unui proces Poisson. Totodată, procesul de servire depinde foarte mult de lungimea pachetelor ce se transmit, deoarece viteza de transmisie pe linie fiind constantă timpul de transmisie al unui pachet este funcție de lungimea sa.

Distribuția timpului de serviciu al unui comutator, depinde de tipul de pachete utilizate de rețea. Dacă rețeaua transmite mesaje în regim de time-sharing, pachetele sînt mici și legea de serviciu destul de aleatoare este asemănătoare unei legi exponențiale. Cînd se realizează transferuri de fișiere, toate pachetele vor avea lungimea maximă de 256 bytes și distribuția timpului de serviciu este foarte apropiată de o lege constantă. În situația în care se suprapun mai multe feluri de trafic, legea va avea o dispersie ridicată și se va apropia de o lege hiperexponențială. Potrivit categoriei de trafic, pătratul coeficientului de variație a distribuției timpilor de serviciu poate oscila de la zero la 2 sau la 3. În figura 3.18 s-a trasat curba timpului de răspuns a unui nod de comutație folosind procedura S—W, pentru q=0,1 și diferite rate de sosire λ.

3.5. CONSIDERAȚII PRIVIND MODELAREA COMPORTĂRII STI CU ȘIRURI DE AȘTEPTARE

Orice analiză matematică a unei probleme de estimare a unor caracteristici a STI prin teoria șirurilor de așteptare reprezintă aproximări ale unor situații reale mult mai complexe. Aproape întotdeauna aplicarea acestor

estimări conduce la obținerea de rezultate „ușor eronate" față de valorile reale precum lungimi mai mari ale șirurilor și întîrzieri mai importante la așteptările din șiruri. Aceste rezultate sînt mai frecvente cînd sînt utilizate sosiri aleatoare și timpi de serviciu aleatori, deviațiile față de valorile reale fiind însă rezonabile din punct de vedere tehnic, asigurînd proiectantului o marjă de siguranță în adoptarea formulelor la modelele mai complexe.

Există însă și situații cînd valorile lungimilor șirurilor și întîrzierilor vor fi mult mai reduse decît cele indicate de curbele trasate. Astfel uneori comportarea normală a sistemului este posibil să se schimbe brusc, cum se întîmplă în cazul sosirii unui număr mare de mesaje la un calculator-nod de comunicație. Alteori timpul de serviciu se mărește mult, în afara limitelor normale, datorită faptului că sistemul gestionează mesaje de lungime mare ce saturează sau blochează liniile de comunicație, bufferele și chiar fișierele de lucru. Din punct de vedere statistic, în ambele situații, apar deci un număr redus de valori ale unor variabile ce sînt mult mai mari decît restul celor uzuale. Așadar media valorilor va fi mică, iar dispersia poate da valori nespecifice. În această împrejurare este posibil să fie evaluate lungimile șirurilor de așteptare, ignorîndu-se timpii de serviciu prea mari.

Uneori presupunerea că timpii intersosiri ai mesajelor urmează o distribuție Poisson nu este corectă. Această eroare nu introduce totuși imprecizii

Dacă se poate totuși presupune că mesajele care intră într-un sistem urmează o distribuție Poisson, acest lucru nu mai este valabil pentru șirul cererilor de acțiuni asupra fișierelor cînd software-ul specific pentru lucrul cu fișiere va avea un efect de filtrare, conducînd la netezirea distribuției evenimentelor generate ca răspuns la cerere. Astfel, într-o configurație cu o bază de date distribuită într-o rețea și la valori de vîrf ale cererilor la un canal de acces la baza de date, timpii intersosiri vor fi constanți, în loc să urmeze o distribuție de tip Poisson.

Tipul de v.a. trebuie ales în funcție de cel al mărimii de evaluat și de modelat, ținînd seama de următoarele:

- timpii de intersosiri și de interplecări ai mesajelor de intrare și de ieșire la o stație de servire, timpii de așteptare pentru sosire, timpii de servire ai mesajelor și timpii de răspuns pentru terminale sînt v.a.c. pozitive;
- numărul mesajelor care sosesc sau pleacă într-un interval T la o stație de serviciu și lungimea șirului de așteptare reprezintă v.a.d. pozitive;
- Pentru stațiile de servire (unitate centrală, echipamente periferice etc.) timpul de gîndire al folosirii acestor resurse în cazul partajării în timp urmează o distribuție exponențială;
- Dacă procesele de sosiri/plecări ale mesajelor la/de la o stație de servire urmează o distribuție Poisson, atunci timpii intersosiri/interplecări sînt descriși de distribuția exponențială.

Pentru o modelare pertinentă a STI actuale tot mai complexe și diverse se indică aplicarea teoriei rețelelor de așteptare deschise în cazul prelucrărilor de tip RJE, pe loturi, transfer de fișiere și a teoriei rețelelor de așteptare închise în cazul STI tranzacționale și a mediilor cu baze de date distribuite (vezi paragraful 6.3).

CAPITOLUL 4

TEHNOLOGIA EALIZĂRII UNUI SISTEM TELEINFORMATIC

Diversificarea echipamentelor de tehnică de calcul, pe de o parte, și apariția unor concepte noi privind arhitectura și structura sistemelor informatice, pe de altă parte, au contribuit la evoluția tehnologiilor informatice. Prin descentralizarea și distribuirea prelucrărilor, datelor și comunicațiilor, în realizarea sistemelor informatice s-au produs mutații cu caracter metodologic datorate apropierii continue a sistemului de calcul de utilizatorul uman.

Condițiile evoluției actuale și de perspectivă a teleinformaticii în țara noastră impun dezvoltarea unei tehnologii de realizare a sistemelor teleinformatice, care să se bazeze și pe metodologia de construire a sistemelor informatice și a produselor-program [T—2].

4.1. NOȚIUNI INTRODUCTIVE

Sistemul teleinformatic are un ciclu de viață care începe în momentul luării deciziei de realizare a acestuia. Urmează realizarea propriu-zisă, apoi exploatarea curentă, împletită cu întreținerea sa (fig. 4.1). Încheierea ciclului se produce în momentul în care sistemul teleinformatic nu mai corespunde cerințelor pentru care a fost conceput sau cînd se impune o modernizare a sa care-i stopează exploatarea curentă.

Datorită complexității sale, realizarea unui sistem teleinformatic reprezintă un efort tehnic, financiar și organizatoric considerabil. Abordarea sistemică, integralistă care se cere în rezolvarea acestei probleme conduce la

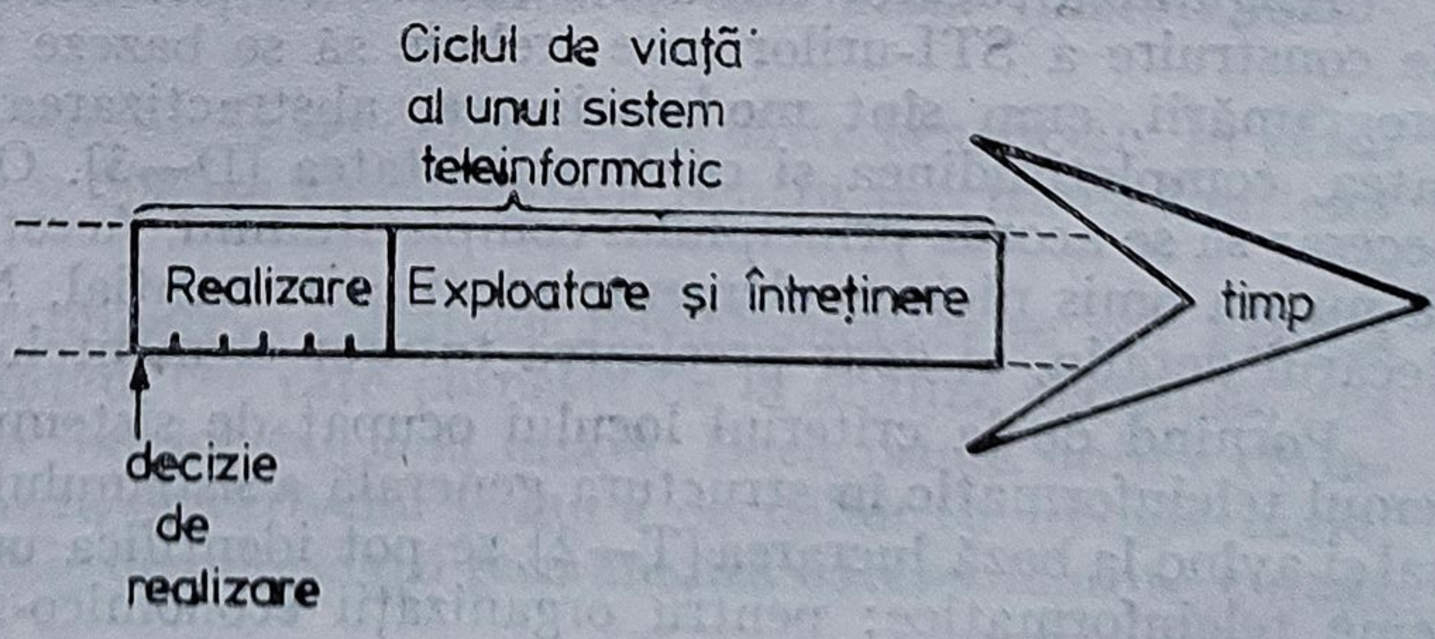


Fig. 4.1. Evoluția în timp a unui STI

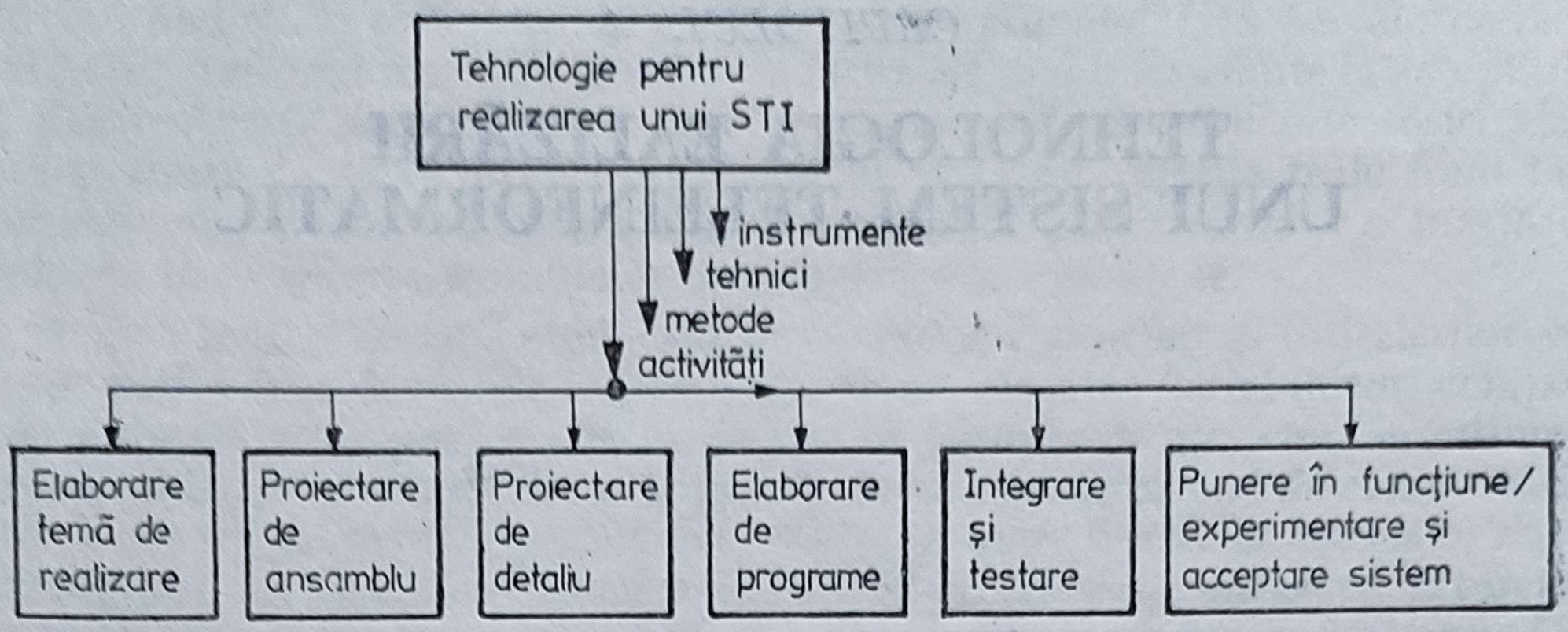


Fig. 4.2. Activități în tehnologia pentru realizarea unui STI

identificarea unor activități, a unor metode, tehnici și instrumente, dintre care unele sînt specifice caracterului impus de accesul la/de la distanță la resursele de calcul și informaționale. Activitățile, metodele, tehnicile și instrumentele alcătuiesc împreună tehnologia de realizare a sistemului teleinformatic.

Realizarea unui sistem teleinformatic implică următoarele activități (fig. 4.2), care corespund fiecare unei etape anumite din ciclul de viață al STI-ului: elaborarea temei de realizare; proiectarea de ansamblu; proiectarea de detaliu; elaborarea de programe; integrarea și testarea; punerea în funcțiune/experimentarea și acceptarea sistemului.

Elaborarea unor metode științifice pentru organizarea și conducerea activităților legate de realizarea în stil industrial a STI-urilor are ca obiectiv final obținerea unei calități garantate, la un preț de cost cît mai redus. Fiecare STI construit trebuie să răspundă calităților de adaptabilitate, eficiență și fiabilitate. Adaptabilitatea dă posibilitatea efectuării fără efort a modificărilor impuse de adăugarea — de exemplu — a unor funcțiuni noi, de îmbunătățirea performanțelor, înlăturarea unor anomalii etc. Eficiența se referă atît la calitatea ansamblului mijloacelor hard și soft de a realiza funcțiile cerute STI-ului în condițiile impuse de sistemul-obiectiv, cît și la efortul financiar necesar. Evitarea, depistarea și înlăturarea erorilor în toate etapele de construire a STI-ului, precum și măsurile pentru restabilirea funcționării corecte a întregului sistem în cazul apariției de erori în timpul exploatării curente sînt atributele fiabilității. În fiecare etapă de realizare a unui STI este important să se țină seama de fiabilitate, pentru ca ea să nu apară ca un simplu adaos în momentul intrării în exploatare.

Asigurarea tuturor calităților enumerate anterior presupune o tehnologie de construire a STI-urilor care trebuie să se bazeze pe principiile inginerieî programării, cum sînt modularizarea, abstractizarea, localizarea, uniformitatea, completitudinea și confirmabilitatea [D—3]. O atenție deosebită este necesar să se acorde principiului completitudinii. Acesta se referă la asigurarea că nu s-a omis nici un element constitutiv esențial. Nu se impune indicarea fiecărui detaliu, ci doar precizarea tuturor elementelor esențiale.

Pornind de la criteriul locului ocupat de sistemul-obiect asistat de sistemul teleinformatic în structura generală a sistemului economico-social național și avînd la bază lucrarea [T—2], se pot identifica următoarele clase de sisteme teleinformatice: pentru organizații economico-sociale complexe (cen-

trale, trusturi, combinate, uniuni cooperatiste etc.); teritoriale; de ramură (aparținînd unor departamente sau ministere); funcționale (destinate unor organe centrale de sinteză, cum sînt CSP, DCS, CNST etc.); ale conducerii superioare de stat.

O clasificare a STI-urilor în funcție de domeniul de utilizare este mai dificil de făcut, deoarece la nivelul fiecărui tip pot fi întîlnite mai multe sfere de aplicație, cum ar fi: informatizarea proceselor de conducere și asistarea deciziei (planificare, coordonare, control și asistarea deciziei); prelucrarea informației economico-sociale; informatizarea documentării și servicii (publice) de informare etc.

Noțiunile introduse în acest capitol vor fi exemplificate pe un sistem teleinformatic destinat comenzii, controlului și reglării unor procese la nivel macro: sistem de tip dispecer național pentru o resursă energetică, de exemplu gazul natural.

4.2. ANALIZĂ PRELIMINARĂ ȘI SPECIFICARE FUNCȚIONALĂ

În vederea realizării unui STI, precizarea sistemului-obiect și a cerințelor de informații ale sistemului de conducere se pot face numai printr-o activitate de *analiză*.

4.2.1. COMPONENTELE ACTIVITĂȚII DE ANALIZĂ

Prin analiza preliminară, care se impune în prima etapă, se urmăresc delimitarea ariei sistemului-obiect, precizarea cerințelor și a restricțiilor în realizarea STI, precum și evidențierea efectelor economice scontate să fie obținute prin noul sistem.

Cunoașterea în detaliu a sistemului-obiect presupune identificarea elementelor sistemului și a legăturilor dintre ele, a intrărilor/ieșirilor sistemului, a stărilor, definirea funcțiilor de transformare a intrărilor în ieșiri și a funcțiilor de transformare a unei stări în alta.

La stabilirea elementelor sistemului-obiect se pot aplica două concepte: primul orientat pe organism (pentru precizarea structurii organizatorice), al doilea dirijat pe activitate (pentru determinarea structurii funcționale).

Activitatea de analiză include culegerea de informații și sistematizarea acestora; un rol important îl deține și evaluarea sistemului informațional-decizional existent.

Sistemul-obiect poate sau nu să fie delimitat de către beneficiar. În cazul în care nu este se impune o analiză suplimentară pentru precizarea limitelor sale.

În activitatea de determinare a cerințelor teleinformatice se pot aborda variante cum sînt, de exemplu, cea care pornește de la analiza deciziilor, cea care pornește de la analiza datelor sau una mixtă. Dacă prima obține cerințele de informații prin analiza obiectivelor și/sau a deciziilor ce trebuie luate, cea de a doua obține cerințele de informații prin analiza datelor folosite curent

în sistemul existent sau a celor potențial folosibile; ultima reprezintă o combinație între primele două.

Activitatea de analiză se realizează prin diferite tehnici care servesc la elaborarea modelului infologic al produsului teleinformatic. Aceste tehnici diferă în funcție de tipul de abordare folosit în determinarea cerințelor informaționale, precum și de modul de realizare a operațiilor care compun activitatea de analiză.

Tehnicile de analiză pot fi elementare sau complexe. Tehnicile elementare de analiză presupun doar operația de culegere de informații și o sistematizare redusă a acestora. Informațiile culese se referă la sistemul informațional-decizional existent și la necesitățile informaționale ale sistemului de conducere. Sistematizarea lor este redusă, deoarece nu sînt sintetizate, ci doar prezintă o aranjare coerentă.

Observarea directă, interviul, chestionarul și studiul documentar (al actelor normative și a regulamentelor de organizare a unităților ce vor fi cuprinse în sistemul teleinformatic, al documentelor din sistemul de evidență a aceloraşi unități) fac parte din tehnicile elementare de analiză. Dintre ele, cea a interviului și a chestionarului sînt utilizate mai des.

Tehnica interviului permite culegerea informațiilor prin discuții — pe bază de întrebări — purtate de către analist cu cadre de conducere și execuție ale beneficiarului viitorului sistem teleinformatic. Întrebările presupun un efort de creativitate. Analistul trebuie să aibă cunoștințe temeinice legate de modalitățile hard și soft care permit accesul la/de la distanță la puterea de calcul și de memorare a sistemelor de calcul, cele care introduc elementele de „tele" (distanță). El va orienta discuția astfel, încît beneficiarul să-și poată defini singur sistemul cel mai eficient pentru sarcinile sale, ajungîndu-se la un compromis între ceea ce inițial se dorea să se realizeze și potențialul hard/soft de implementare efectivă a sistemului teleinformatic.

Formularea în scris de către analist a întrebărilor conduce la tehnica chestionarului, care nu poate înlocui interviul, ci doar îl completează. Chestionarul este însoțit, de regulă, de instrucțiuni privind modul de completare. În activitatea de analiză sînt utilizate chestionare tematice (cu o singură temă sau cu mai multe teme), de conținut (cu date factuale sau de opinie), cu întrebări închise (care presupun selectarea unui răspuns dintr-o serie de variante) sau cu întrebări deschise.

În etapa de elaborare a temei de realizare cu ajutorul tehnicii chestionarului se poate sonda opinia referitoare la oportunitatea introducerii sistemului teleinformatic. La baza construirii chestionarelor este posibil să stea principii de natură tehnică, principii de natură psihologică etc.

Tehnicile complexe de analiză realizează culegerea și sistematizarea informațiilor, precum și evaluarea sistemului informațional-decizional existent. Sistematizarea informațiilor, care implică și o sintetizare, se efectuează folosind scrierea tehnică, tehnica diagramelor, a grilelor sau tabelele de decizie. Evaluarea sistemului informațional-decizional vizează: indicatorii specifici prelucrării datelor, coeficienții de estimare a prelucrărilor efectuate în cadrul sistemului teleinformatic, separînd prelucrările necesare transportului fără erori a datelor în întregul sistem și asigurarea livrărilor la destinație a datelor în intervalul de timp impus. Tehnicile complexe de analiză pot fi completate cu instrumente specializate care să permită analiza asistată de calculator.

4.2.2. TEMA DE REALIZARE

Documentația care înglobează rezultatele activității de analiză preliminară și specificare funcțională o reprezintă tema de realizare. În introducere tema de realizare trebuie să cuprindă baza de elaborare a sistemului teleinformatic și anume să specifice documentațiile în temeiul cărora se face elaborarea lui precum și organizația care a aprobat elaborarea și eventual data aprobării.

Capitolul dedicat cerințelor și restricțiilor trebuie să analizeze global sistemul-obiect, să prezinte sistemul nou propus, cadrul tehnologic de realizare și de control al calității, precum și necesarul de resurse și termenele estimative pentru realizarea sistemului teleinformatic. Partea destinată analizei globale a sistemului-obiect descrie obiectivele generale, organizarea, funcțiile și relațiile interne și externe, aspectele critice de soluționat, cum ar putea fi cele legate de asigurarea unui optim în partea de comunicații. De asemenea nu este lipsit de interes să se cunoască stadiul informatizării activităților din unitățile beneficiarului implicate în STI, cu evidențierea aplicațiilor existente, a dotării și a personalului specializat. Colecțiile de date, fluxurile, principalele procese de prelucrare, interconexiunile, aspectele dimensionale și dinamice descriu global procesele informatizabile. Utile pot fi și funcțiile, activitățile și procesele neinformatizabile.

Prezentarea sistemului nou propus trebuie să cuprindă aria de aplicabilitate, domeniile și problemele acoperite, obiectivele. Componentele funcționale, relațiile între componente, performanțele estimate conturează modelul funcțional global al sistemului teleinformatic. Soluții de organizare și gestionare a datelor se impun.

În partea rezervată arhitecturii hard/soft este necesar să fie folosite acele soluții care să satisfacă cît mai bine cerințele beneficiarului, dar să introducă un număr cît mai mic de elemente de cuplare. Resursele necesare instalării și exploatării sistemului teleinformatic vor cuprinde echipamente, software-ul de bază, de comunicații și cel de aplicații, personalul implicat, amenajările de spații impuse de noul sistem. Pot fi prevăzute și eventuale extensii în timp, care — de obicei — vizează volume noi de date și funcțiuni noi. Condițiile de acceptare a sistemului vor încheia capitolul referitor la prezentarea sistemului teleinformatic nou propus.

Un capitol deosebit de important în tema de realizare este cel care reunește cerințele privind documentația finală a sistemului teleinformatic, etapele de realizare, documentațiile intermediare, punctele de control, verificările și validările de efectuat, procedura de recepție.

Necesarul de resurse și termenele estimative pentru realizarea sistemului teleinformatic, cu precizarea manoperei, a capacității de calcul, a eșalonării termenelor reprezintă următorul capitol al temei de realizare. Dacă comunicațiile între sistemele de calcul se fac avînd la bază linii de telecomunicație trebuie prevăzute resurse financiare pentru închirierea acestora încă din primele etape ale realizării sistemului teleinformatic. Aceasta apare ca o mențiune separată din cauza tarifelor încă ridicate, oferite de serviciile de comunicații telefonice.

Capitolul care încheie tema de realizare cuprinde justificarea necesității și oportunității realizării sistemului. Cheltuielile pentru elaborarea, insta-

larea, punerea în funcțiune, pregătirea personalului și exploatarea sistemului teleinformatic trebuie luate în calcul. Față de exploatarea în local, exploatarea sistemului teleinformatic introduce o nouă dimensiune în complexitate datorată multiplelor sisteme de calcul interconectate. Este important să se precizeze și termenele propuse pentru recuperarea cheltuielilor.

4.2.3. STUDIU DE CAZ

Să luăm în considerare un dispecerat care la nivel național are sarcina de a supraveghea activitatea de extracție a gazului metan pînă la nivelul cîmpurilor gazeifere; rețeaua de conducte magistrale prin care se realizează transportul gazului metan și distribuția gazelor la nivel de mari consumatori.

Utilizarea unui sistem teleinformatic care să asiste activitatea de dispecerizare ar avea ca principal scop obținerea de performanțe în realizarea unui consum rațional al gazului natural la nivelul întregii țări.

Creșterea cantității și calității informațiilor colectate și prelucrate în sistem, precum și crearea premiselor pentru utilizarea unor modele de simulare a procesului de transport și distribuție a gazelor (rezolvabile ușor în condițiile existenței unui sistem teleinformatic corespunzător) contribuie la obținerea unei calități superioare în activitatea de dispecerizare. Culegerea de informații permite cunoașterea organizării sistemului, a fluxului activităților și a fluxului de informații. Funcționarea sistemului este structurată pe mai multe niveluri, după cum urmează:

- nivelul de bază de culegere a parametrilor, format din puncte de măsurare a gazelor, stații de reglare și măsură, stații de comprimare a gazelor, toate reunite sub denumirea de puncte local controlate;
- nivelul intermediar de colectare a parametrilor tehnologici culeși de la punctele local controlate, amplasat în cadrul unor dispecerate regionale de întreprindere;
- nivelul dispeceratelor regionale care colectează informații de la dispeceratele regionale de întreprinderi;
- nivelul centralizat de prelucrare a informațiilor, cuprinzînd centrul de prelucrare centralizată a datelor;
- nivelul decizional, amplasat la dispeceratul național de gaze naturale.

Ierarhia stabilită în sistemul-obiect este prezentată în fig. 4.3.

Funcțiile sistemului-obiect ce trebuie precizate în primul rînd pot cuprinde: contabilizarea zilnică a gazelor extrase, pe furnizori, schele, brigăzi, structuri gazeifere de înaltă și joasă presiune; urmărirea și controlul operativ al presiunilor tehnologice în rețeaua de transport și distribuție; realizarea operativă a bilanțurilor de gaze; urmărirea și calculul debitelor de gaze; supravegherea temperaturilor mediului ambiant pe zone de consum; elaborarea și actualizarea repartițiilor de gaze pe consumatori, cu verificarea încadrării în capacitățile de producție, transport și distribuție a gazelor; controlul încadrării în repartițiile de gaze ale beneficiarilor; corelarea consumurilor de gaze în funcție de condițiile meteorologice.

Pasul următor în activitatea de analiză îl reprezintă formularea cît mai riguroasă a cerințelor și restricțiilor față de noul sistem. Se poate formula

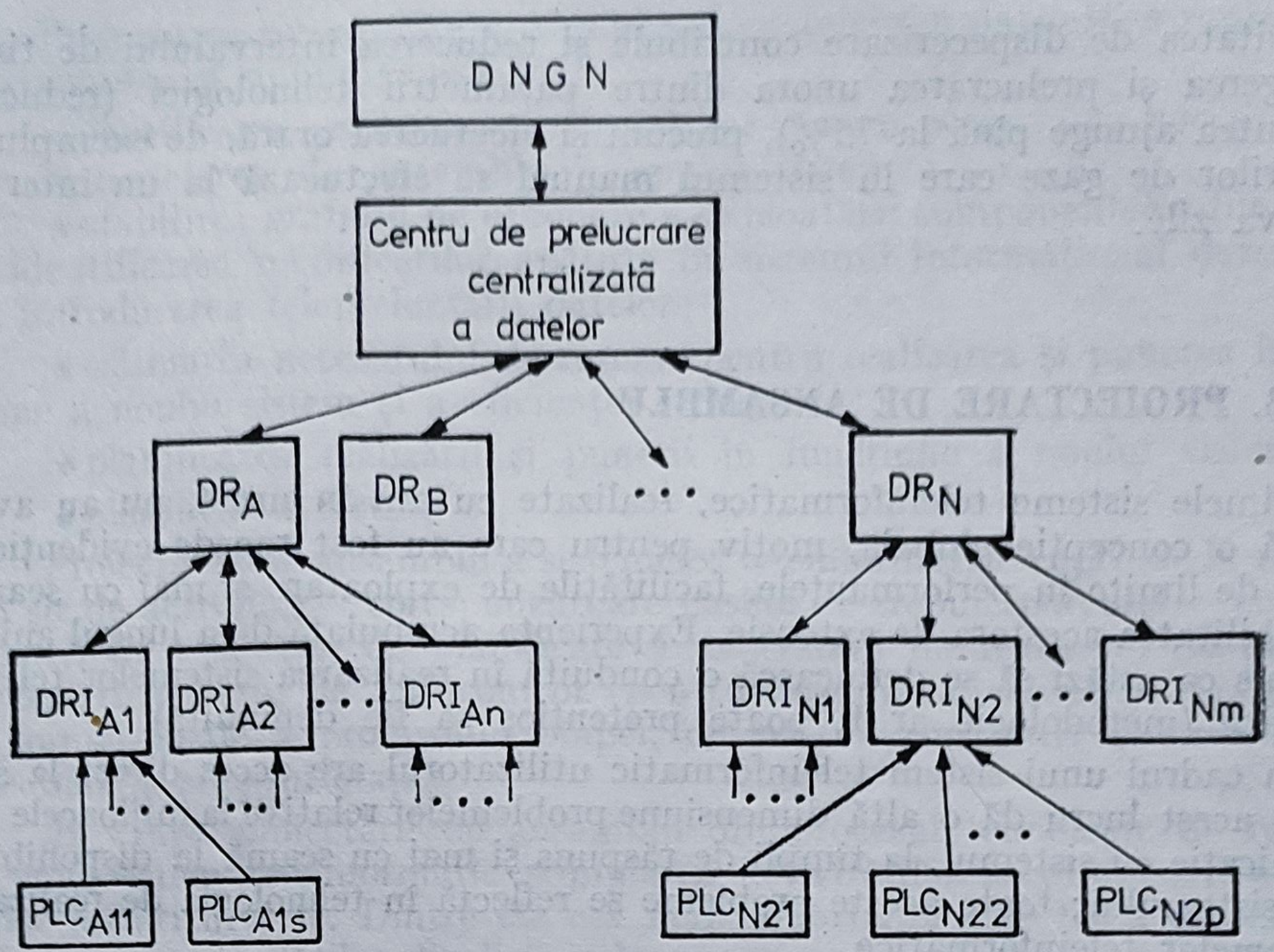


Fig. 4.3. Ierarhia nivelurilor în sistemul-obiect al cazului luat în studiu: *DNGN* — Dispecerat Național de Gaze Naturale; *DR* — Dispecerat Regional; *DRI* — Dispecerat Regional de Întreprindere; *PLC* — Punct Local Controlat.

ca cerință furnizarea de către sistemul teleinformatic a anumitor informații, din care unele să fie prezentate la fiecare oră (de exemplu, presiunile tehnologice în rețeaua de transport și distribuție gaze naturale), altele la 6 ore (debitele preluate din sistem de către consumatori), la 24 ore (producția de gaz metan pe furnizori) și, în sfîrșit, informații cu periodicitate mai mare de 24 de ore (date despre consumatori și furnizori). Periodicitatea este justificată prin deciziile pe care le susțin informațiile respective.

Restricțiile de natură informatică care intervin prevăd utilizarea pe cît posibil a unor baze informatice deja existente la beneficiar și amplasate cît mai aproape de dispeceratele regionale. De asemenea, interconectarea dispeceratelor regionale trebuie să se facă de așa manieră, încît să se permită mesajelor ce vor fi vehiculate prin sistemul teleinformatic căi multiple între două noduri ale rețelei de calculatoare, în condițiile unei valori cît mai mici a lungimilor tuturor liniilor de comunicații. Se știe că scurtarea liniilor de comunicație atrage după sine reducerea cheltuielilor de exploatare.

Ultima problemă care interesează în etapa de analiză este cea referitoare la efectele economice scontate. Se apreciază că prin introducerea sistemului teleinformatic în asistarea dispecerizării la nivelul unei țări a gazului metan se obține o reducere substanțială (de cca *n* ori) a volumului de muncă și aceasta pe seama ridicării considerabile a calității acesteia. Volumul de informații preluat și prelucrat în sistemul teleinformatic poate crește de *m* ori față de volumul de informații manipulat anterior; puterea de calcul și de memorare a sistemelor de calcul cuprinse în rețeaua de calculatoare conduce la realizarea unor prelucrări complexe. La obținerea unei calități superioare

în activitatea de dispecerizare contribuie și reducerea intervalului de timp la culegerea și prelucrarea unora dintre parametrii tehnologici (reducere ce ar putea ajunge pînă la 75%), precum și efectuarea orară, de exemplu, a bilanțurilor de gaze care în sistemul manual se efectuează la un interval de cîteva zile.

4.3. PROIECTARE DE ANSAMBLU

Primele sisteme teleinformatice, realizate cu ani în urmă, nu au avut la bază o concepție globală, motiv pentru care au fost repede evidențiate o serie de limite în performanțele, facilitățile de exploatare și mai cu seamă în posibilitatea acestora de extensie. Experiența acumulată de-a lungul anilor a permis ca astăzi să se definească o conduită în realizarea sistemelor teleinformatice (metodologie ar fi poate pretențios să fie denumită).

În cadrul unui sistem teleinformatic utilizatorul are acces direct la sistem și acest lucru dă o altă dimensiune problemelor relative la mijloacele de comunicație cu sistemul, la timpii de răspuns și mai cu seamă la disponibilitatea sistemului; toate aceste probleme se reflectă în tehnologia de realizare a sistemelor teleinformatice.

4.3.1. TEHNOLOGIA ETAPEI

Proiectarea de ansamblu a unui sistem teleinformatic urmărește ca principal scop stabilirea unei liste complete a criteriilor de alegere relative la sistemul teleinformatic, criterii ce trebuie să conducă la soluția care să satisfacă cel mai bine obiectivele din tema de realizare. Aceste criterii pot fi:

- de performanță (timpii de răspuns ai sistemului, întîrzierea datorată traversării prin rețeaua de comunicații, stabilitatea sistemului, extensibilitatea sistemului, capacitatea sistemului de-a satisface apelurile, securitatea informațiilor, disponibilitatea sistemului);
- tehnice (existența echipamentelor, conformarea acestora la normele internaționale, compatibilitatea, posibilitățile de extensie, întîrzieri în livrare sau fabricație, fiabilitatea hardware, existența software-ului de exploatare, fiabilitatea software, supraîncărcarea dată de software, instrumente pentru sprijinirea dezvoltării software-ului, instrumente pentru teste software);
- de exploatare (simplitatea exploatării, politica de întreținere, omogenitatea sau heterogenitatea componentelor, responsabilitatea lăsată în seama operatorilor);
- relative la personal (de calificare a personalului, de stabilitate a sa, politica de recrutare și de formare);
- geografice (prezența personalului de întreținere în toate nodurile rețelei, disponibilitatea pieselor de schimb);
- economice (costul punerii în funcțiune a sistemului, costul exploatării, raportul cost/performanță pe termen scurt, mediu sau lung);
- politice (soluții de prestigiu, furnizor unic).

Proiectarea de ansamblu a unui sistem teleinformatic își are obiectivele ei concrete, și anume:

- specificarea cerințelor și restricțiilor pentru proiectarea noului sistem;
- modelarea de ansamblu a noului sistem;
- stabilirea grafului de ordonare a exploatării componentelor funcționale și identificarea modificărilor apărute în sistemul informațional determinate de introducerea teleprelucrării datelor;
- estimarea necesarului de resurse pentru realizarea și punerea în funcțiune a noului sistem și a eficienței economice;
- planificarea realizării și punerii în funcțiune a noului sistem;
- planificarea testării.

Proiectarea de ansamblu a sistemelor teleinformatice cuprinde la rîndul ei un șir de activități — corespunzătoare pașilor ce trebuie parcurși — cum sînt: tehnici de realizare, de planificare a procesului de realizare, de pregătire a testării/testare, de autocontrol (pentru asigurarea calității), de realizare a formei finale a produsului etapei, de control (normativ și metodologic) efectuat la sfîrșitul etapei.

În etapa proiectării de ansamblu din procesul tehnologic de realizare a unui sistem teleinformatic stimularea activităților se obține cu ajutorul unor intrări diverse. Dintre ele cea mai importantă este tema de realizare. Efectuarea activităților din lista celor menționate mai sus conduce la o serie de produse intermediare, care reprezintă ieșirile; parte din ele au însemnat — la momente anterioare de timp — intrări. Pentru proiectarea de ansamblu, legăturile dintre intrări-activități-ieșiri sînt date în tabelul 4.1.

În etapa proiectării de ansamblu, activitățile tehnice de realizare, care dețin ponderea cea mai mare includ: analiza globală și specificarea cerințelor, modelarea de ansamblu a sistemului propus, structurarea datelor și indicarea soluțiilor de administrare a datelor, proiectarea arhitecturii configurației de echipamente, stabilirea grafului de ordonare a exploatării componentelor funcționale, identificarea modificărilor apărute în sistemul informațional determinate de introducerea teleprelucrării, estimarea necesarului de resurse pentru realizarea și punerea în funcțiune a noului sistem, evaluarea eficienței economice.

În cadrul analizei globale se completează și se definitivează informațiile ce vor fi reținute cu privire la:

- implantarea geografică a unităților economice implicate în sistemul-obiect și participante la sistemul teleinformatic; se precizează poziția aproximativă a locurilor unde o parte din informația întregului sistem este produsă, prelucrată și utilizată, precum și fluxul logic dintre acestea;
- natura prelucrărilor bazate pe calculator, deja existente în unitățile economice implicate;
- existența eventuală a unei infrastructuri teleinformatice;
- caracteristicile informațiilor (natura lor și suporții pe care se găsesc, organizarea și legăturile dintre ele).

Analiza generală a cerințelor permite să se definească la scară macroscopică aplicațiile care se vor realiza în sistemul teleinformatic. Un studiu general al aplicațiilor asigură definirea funcțiilor acestora și stabilirea legăturilor dintre ele.

TABELUL 4.1. Activități (asociate cu intrări și ieșiri) în proiectarea de ansamblu

Intrări / Ieșiri	Specificația cerințelor și restricțiilor externe pentru proiectarea sistemului tehnic informatic	Modelul funcțional al noului sistem	Modelul conceptual al datelor și soluția de administrare a datelor	Arhitectura configurației de echipamente/sisteme	Graful de ordonanțare a exploatării componentelor funcționale	Cerințe privind asigurarea informațională pentru funcționarea noului sistem	Necesar de resurse pentru realizarea și punerea în funcțiune a noului sistem	Eficiența economică	Planul realizării și punerii în funcțiune a noului sistem	Plan de testare sistem	Plan de testare integrare sistem	Specificația de testare pentru testul de acceptanță	Specificația de testare pentru testul de sistem	Specificația de testare pentru testul de integrare sistem	Produse intermediare (validate intern)	Proiect de ansamblu	Raport de evaluare a etapei
Tema de realizare	Activități tehnice de realizare								Activități de planificare a procesului de realizare			Activități de pregătire a testării/ testare			Activități de autocontrol	Realizarea formei finale a produsului etapei	Activități de control efectuate la sfîrșitul etapei
Plan de testare acceptanță																	
Plan de testare sistem																	
Plan de testare integrare sistem																	
Produse intermediare																	
Produse intermediare																	
Proiect de ansamblu																	

Specificarea cerinţelor pentru proiectarea noului sistem vizează elaborarea de soluţii fără să se intre în detaliile de funcţionare ale sistemului sau să se precizeze concret componentele hardware şi software.

Natura şi importanţa sistemului-obiect pot să orienteze alegerile în soluţiile care se au în vedere.

Descrierea externă a sistemului teleinformatic — aşa cum se preconizează a fi — cuprinde: descrierea întregului context organizaţional pe care se sprijină, analiza funcţională a componentelor şi schema generală de funcţionare pentru întregul sistem.

Sistemul întreg se poate descompune (sub aspect funcţional) într-un anumit număr de proceduri funcţionale, fiecare cuprinzînd un eveniment care o declanşează, multiple operaţii pentru prelucrarea acestui eveniment şi unul sau mai multe rezultate furnizate în urma efectuării prelucrărilor.

Procedura funcţională dintr-un sistem teleinformatic implică un număr de schimburi conversaţionale, ce dau naştere la schimbul unui anumit număr de mesaje (intrări şi ieşiri) cu sistemul. Datele de intrare şi recuperarea altor date rezultate din sistem se manipulează cu ajutorul posturilor de lucru.

Analiza funcţională a sistemului, care cuprinde studierea posturilor de lucru, a schimburilor conversaţionale pentru fiecare procedură funcţională, a datelor necesare stabilirii traficului şi a dimensionării sistemului, ajută la modelarea de ansamblu a sistemului propus. În acest moment se poate defini sau redefini repartizarea prelucrării datelor în sistemul teleinformatic.

În proiectarea de ansamblu cercetarea soluţiilor posibile pentru arhitectura configuraţiei de echipamente (care să răspundă temei de realizare) presupune determinarea unei filosofii de reţea şi trebuie legată de tipul de structură al organizării unităţilor economice implicate în sistemul obiect. Figura 4.4 prezintă principalele tipuri de structuri ale unei organizări. Schemele a.0 şi b.0 din figură reprezintă cele două structuri fundamentale: organizarea centralizată şi organizarea repartizată; ele constituie primul nivel de structurare. Structurile fundamentale dau naştere la alte structuri, ajungîndu-se la un al doilea nivel de structurare (schemele a.1, a.2, b.1, b.2). Se poate merge în continuare cu un al treilea, al patrulea, ..., nivel de structurare, obţinînd noi tipuri de structuri.

Reţeaua care reprezintă structura unei organizaţii, adică unităţile acestei organizaţii şi dependenţa dintre ele, se numeşte reţea organizaţională.

Aplicaţiile teleinformatice nu privesc uneori toate unităţile unei organizaţii, ci doar o parte din ele; în aceste cazuri se poate vorbi de subreţele organizaţionale. Deci o reţea organizaţională poate suporta mai multe aplicaţii teleinformatice care utilizează entităţile organizaţiei.

Pornind de la reţeaua organizaţională, în urma unei analize funcţionale se ajunge la o structură ce reprezintă repartiţia datelor, repartiţia prelucrărilor şi comunicaţiile stabilite între unităţile de prelucrare şi de gestiune a datelor şi accesul de la distanţă. Această structură constituie reţeaua funcţională. Există întotdeauna mai multe reţele funcţionale posibile pentru o anumită reţea organizaţională. De exemplu cu reţeaua organizaţională din figura 4.4 — b.0. se pot obţine trei reţele funcţionale, aşa cum apar ele în figura 4.5.

Pe baza reţelei funcţionale se defineşte o reţea organică reprezentînd structura obţinută cu mijloacele informatice care asigură gestiunea datelor şi prelucrarea lor, precum şi mijloacele utilizate la transmiterea datelor între aceste mijloace informatice.

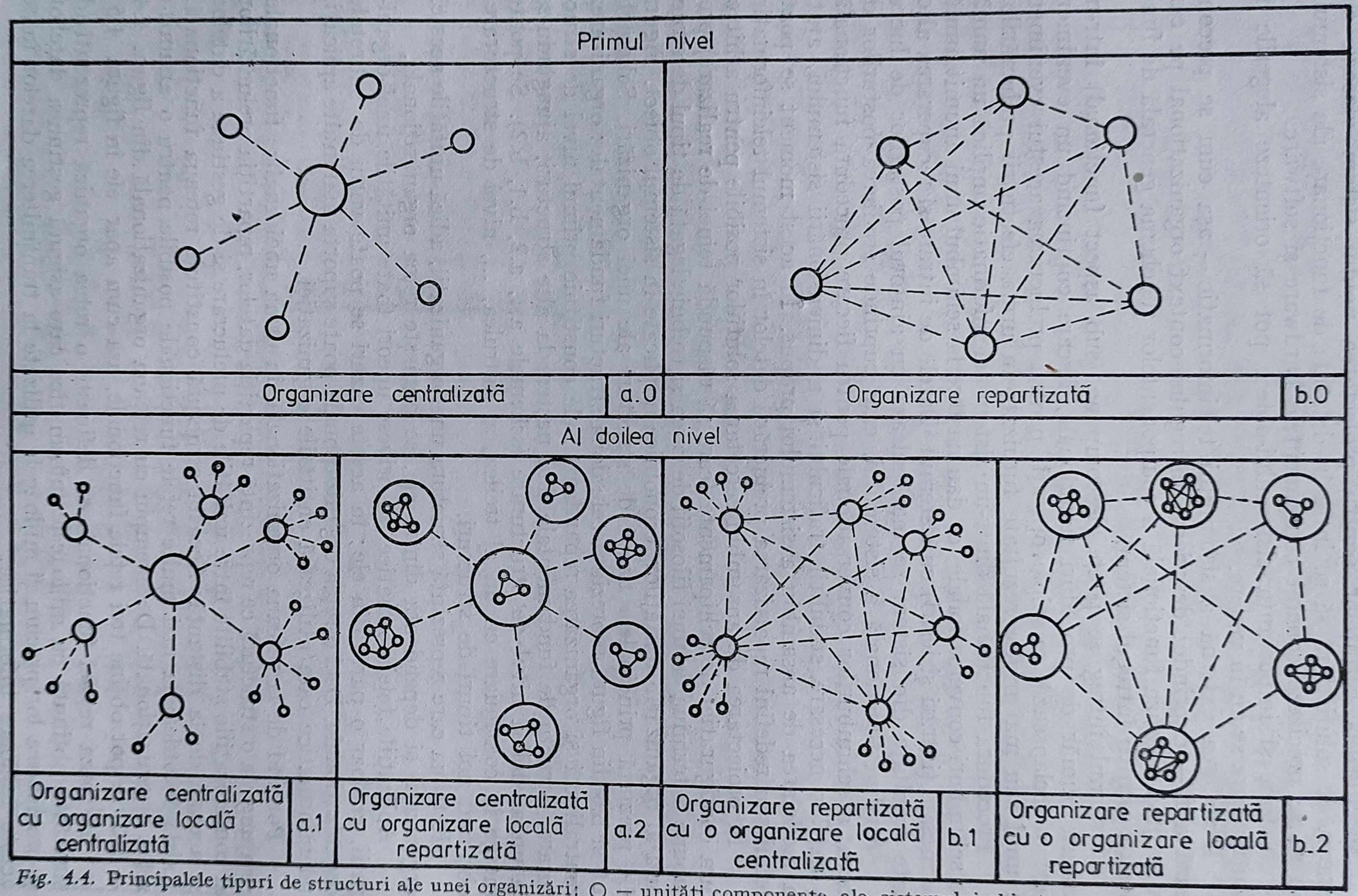
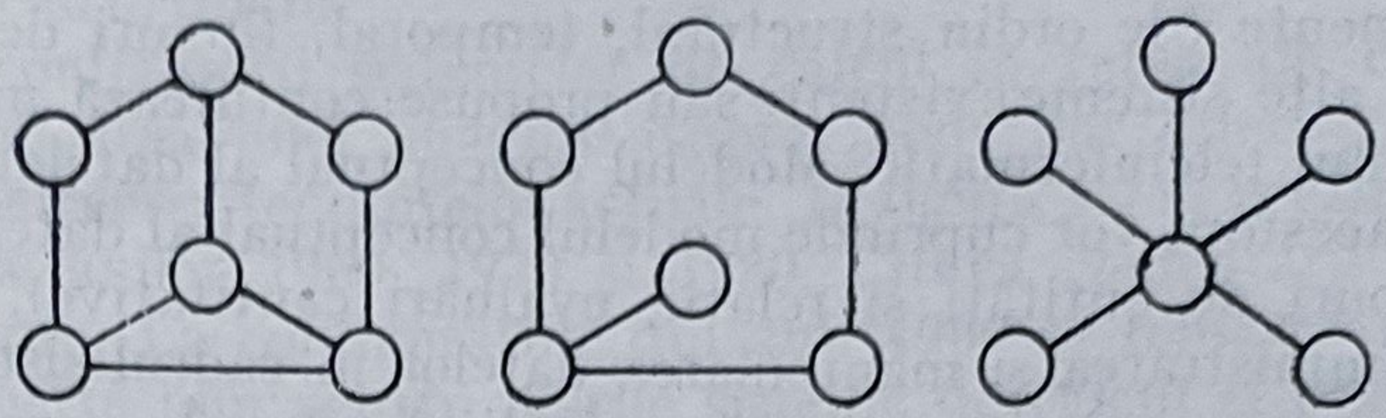


Fig. 4.4. Principalele tipuri de structuri ale unei organizări; ○ — unități componente ale sistemului-obiect; — — — legături de dependență care indică eventual și circulația informațiilor între entități.

Fig. 4.5. Exemple relative la structuri de rețele funcționale



Cum rețelele organizaționale, funcționale și organice se deduc unele din altele, o reprezentare sugestivă a relației dintre ele poate fi cea din figura 4.6.

În etapa proiectării de ansamblu, tehnologia de realizare a sistemului teleinformatic folosește metode, tehnici și instrumente variate, dintre care menționăm: interviul, chestionarul, analiza diagnostic, analiza celulară, modelarea sistemului orientată pe obiecte, proiectarea top-down; proiectarea structurată, proiectarea compusă, tehnica de specificare a interfeței cu utilizatorul folosind gramatici bi-partidă. Pentru partea de arhitectură a configurației de echipamente se recomandă folosirea îndrumarului de echipamente de tehnică de calcul produse în Republica Socialistă România.

Documentația care înglobează rezultatele activității de proiectare de ansamblu și constituie forma finală a produsului etapei o reprezintă proiectul de ansamblu. El trebuie să cuprindă în introducere baza de elaborare, adică să specifice documentele în temeiul cărora se face elaborarea lui, organizația care a aprobat elaborarea, precum și data aprobării. Capitolul dedicat specificației cerințelor și restricțiilor pentru proiectarea noului sistem teleinformatic prezintă următoarele cerințe: informaționale (funcții prioritare, aspecte critice de soluționat, interfața cu utilizatorul/mediul sau cu alte sisteme), globale de integrare (informațională, tehnică și de comunicație), de performanță și de calitate, privitoare la regimul de funcționare, de asigurare a nivelului de calitate dorit. Restricțiile sînt date de respectarea standardelor, limitările hardware, de condițiile externe impuse de sistemul informațional existent.

În capitolul rezervat modelului de ansamblu al sistemului teleinformatic se vor menționa modelul funcțional al noului sistem, modelul conceptual al datelor și soluția de administrare a datelor, arhitectura configurației de echipamente și o formă preliminară a grafului de ordonare al exploatării componentelor funcționale.

Componentele funcționale (structurate și descrise pe niveluri ierarhice și/sau subsisteme, pînă la nivel de aplicații individuale), relațiile între com-

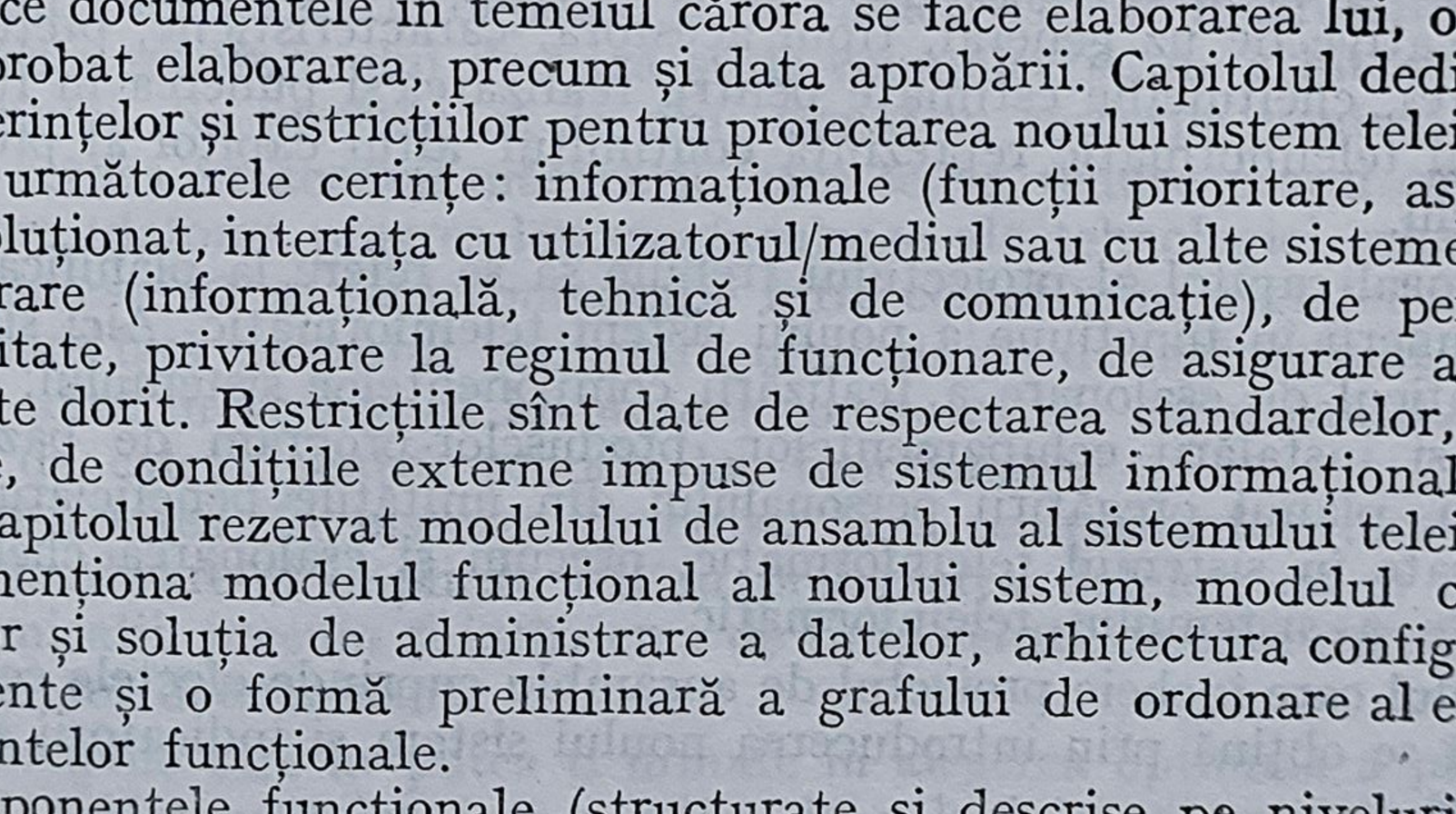


Fig. 4.6. Relația rețea organizațională, funcțională și organică

ponente (de ordin structural, temporal, fluxuri de date), relațiile/interfețele cu alte sisteme existente sau propuse conturează modelul funcțional al sistemului teleinformatic. Modelul conceptual al datelor și soluția de administrare a acestora vor cuprinde modelul conceptual al datelor (colecții/seturi de date, tipuri de entități și relații, evaluări cantitative), soluțiile propuse pentru administrarea și manipularea datelor în cadrul sistemului și cerințele de asigurare a corectitudinii și securității datelor. În paragraful destinat arhitecturii configurației de echipamente se vor preciza tipurile de echipamente și configurațiile, mijloacele de comunicație necesare (terminale, linii, modemuri, concentratoare, testoare), condițiile și punctele de amplasare, cerințele/restricțiile speciale de instalare, precum și amenajările necesare (spații, săli, clădiri, instalații). Frecvențele, termenele de execuție și relațiile de precedență/succesiune între componentele funcționale vor încheia capitolul.

Problematica reorganizării fondului de date existent la utilizator și implicațiile organizatorice ale introducerii teleinformaticii constituie capitolul dedicat cerințelor privind asigurarea informațională în funcționarea sistemului teleinformatic.

Necesarul de resurse pentru realizarea sistemului teleinformatic (capacitate de calcul și de comunicații, manoperă, alte resurse), cel de punere în funcțiune a sistemului (necesar de personal și lista de echipamente, tipul acestora, caracteristicile de catalog, cantitatea, prețul, sursa de procurare, precum și lista de produse-program de bază, cele destinate comunicațiilor și cele aplicative de uz general, tipul acestora, caracteristicile, prețul, sursa de procurare), cheltuielile estimate pentru realizarea și punerea în funcțiune a sistemului teleinformatic reprezintă conținutul altui capitol al proiectului de ansamblu.

Următorul capitol al proiectului trebuie să se refere la planificarea realizării și punerii în funcțiune a noului sistem teleinformatic. Aici sînt menționate graficul de eșalonare a realizării componentelor sistemului, graficul procurării și instalării echipamentelor, produselor-program de bază și de comunicații, planul pregătirii personalului din unitățile beneficiarului care sînt implicate în sistemul teleinformatic, precum și eșalonarea cheltuielilor de realizare a sistemului teleinformatic.

Capitolul care încheie proiectul de ansamblu cuprinde efectele economice scontate să se obțină prin introducerea noului sistem și indicatorii sistemici de eficiență.

4.3.2. ASPECTE PRIVIND PROIECTAREA STRUCTURILOR PENTRU REȚELE

În proiectarea structurilor pentru rețelele de calculatoare soluțiile se aleg dintr-un spectru larg de posibilități, fiecare din posibilități prezentînd avantaje unice pentru o anumită utilizare. La motivarea din punct de vedere tehnologic a soluției alese (de exemplu cu privire la tipurile de calculatoare ce urmează să fie folosite: calculatoare mari, minicalculatoare, calculatoare personale) se adaugă și alte considerații. De multe ori argumentarea alegerii tipului de calculator se bazează pe considerații de cost. Din punctul de vedere al tehnologiei și al gradului de dezvoltare avantajele pe care pare să le ofere un anume tip de calculator glisează de la o clasă la alta de calculatoare. Sînt

multe cazurile în care considerente de ordin organizatoric și, mai ales, decizional impun anumite calculatoare.

Parcurgerea istoriei configurării sistemelor de calcul în rețele de calculatoare [I—1] permite crearea unei imagini asupra modurilor diferite de utilizare, cunoașterea și înțelegerea tendințelor actuale. În figura 4.7 se prezintă configurațiile caracteristice anilor 1960, 1970 și 1980. Cîteva comentarii pe marginea lor sînt interesante. La configurațiile din anii 1960 se remarcă raportul cost/performanță foarte bun pentru marile sisteme de calcul și costul mai mic al comunicațiilor decît cel al prelucrărilor. În anii 1970 se observă că raportul cost/performanță este bun pentru minicalculatoare (care au capacitatea marilor sisteme din decada precedentă); costul comunicațiilor devine mare datorită atît a folosirii capacității de prelucrare a calculatoarelor și pentru comunicații, cît și a introducerii și utilizării tot mai mult a displayurilor (care cer lărgimi de bandă mari) ca terminale. Specific anilor 1980 este în primul rînd gama largă de capacități de prelucrare disponibile, la ele contribuind și calculatoarele personale (ce au capacitatea marilor sisteme din decada anilor 1960); costul comunicațiilor continuă să devină ridicat, în parte ca urmare a folosirii interfețelor utilizator avansate, care necesită lărgimi de bandă foarte mari.

Care sînt tendințele în prezent? Exprimată în termeni de milioane de instrucțiuni executate pe secundă, în megaocteți de memorie internă sau memorie disc, capacitatea hardware trebuie plasată de așa manieră în rețea, încît să se obțină un raport cost/performanță cît mai bun. Productivitatea sistemului teleinformatic depinde și de caracteristicile încărcării cu aplicații, de eficiența software-ului de bază și de comunicație, de arhitectura de bază a sistemului. Se prevede că în anii următori costurile pentru comunicații nu vor scădea în aceeași măsură în care se va reduce costul capacității de prelucrare (datorat progreselor spectaculoase ale tehnologiei calculatoarelor). Se obișnuiește să se facă comparație între costul unui apel telefonic pe distanță mare față de costul prelucrării unui milion de instrucțiuni pe secundă. Trebuie avut în vedere că în prezent comunicațiile cer o cantitate mare de resurse de calcul pentru propria lor funcționare.

Într-o rețea, în plasarea funcțiilor și a echipamentelor, costul necesar pentru hardware-ul implicat atît în prelucrarea datelor, cît și cel implicat în comunicații, precum și întîrzierea de timp cerută de executarea aplicațiilor joacă un rol determinant.

Principiile care ar putea fi urmate în alegerea optimală a plasării funcțiilor și echipamentelor într-o rețea de calculatoare vizează interfața cu utilizatorul, aplicațiile, precum și proiectarea rețelei de comunicații. Utilizatorul solicită astăzi interfețe puternic interactive cu sistemul teleinformatic (ce implică creion luminos pentru selectarea meniului, grafică multicromatică etc.). Aceste interfețe se realizează cu lărgimi de bandă ridicate pentru comunicații. De asemenea se impun timpi de răspuns tot mai scăzuți. La toate aceste cerințe tehnica clasică, bazată pe TTY/displayuri conectate la distanță față de sistemul de calcul, nu corespunde. Soluția o reprezintă utilizarea stațiilor de lucru grefate pe microprocesoare în care să se implementeze interfața cu utilizatorul.

Pentru a evidenția principiile ce vizează aplicațiile în alegerea optimală a plasării funcțiilor și echipamentelor într-o rețea de calculatoare se dă un exemplu menit să sublinieze relațiile între plasarea programelor de aplicație

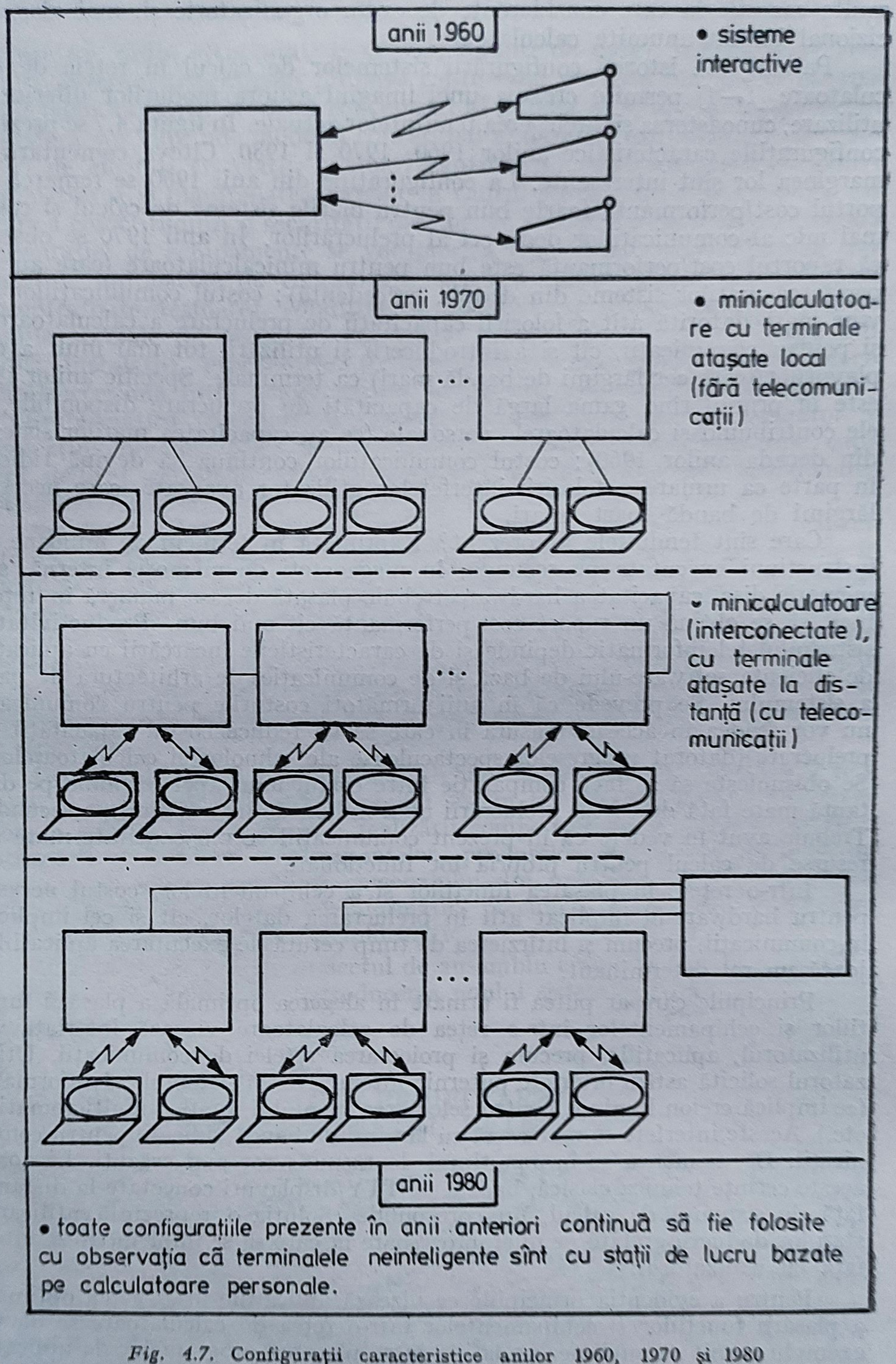


Fig. 4.7. Configurații caracteristice anilor 1960, 1970 și 1980

și datele asociate acestora. Tabelul 4.2, prin datele prezentate, dă o imagine a ceea ce înseamnă executarea unei aplicații cu date asociate și existente în același sistem de calcul sau folosind date aflate la depărtare pe alt sistem de calcul. În cazul cînd datele sînt furnizate local se accesează discul pe care sînt memorate acestea. Aplicația primește un mesaj de la utilizator, accesează discul pentru a lua date necesare prelucrării și, în urma prelucrărilor, utilizatorului i se trimite un mesaj de ieșire. Fiecare din funcțiile menționate mai sus sînt exprimate în tabel prin număr de instrucțiuni necesare a fi executate.

TABELUL 4.2 Evidențierea plasării funcțiilor și echipamentelor față de sursa de date într-o rețea de calculatoare

Aplicația		Amplasarea sursei de date	Funcțiile realizate	Numărul de instrucțiuni (mii de instrucțiuni)	
prezintă facilități teleinformatice	locul execuției			pentru o funcție	total
nu	local	local	• recepție mesaj 5 • accese la disc (zece) 50 • trimiterea unui mesaj de ieșire 5 • aplicația 100		160
da — nodul local	local	la distanță	• recepție mesaj 5 • trimite mesaje (zece) pentru stocare disc 50 • recepție mesaje (zece) cu înregistrări de pe disc 50 • trimiterea unui mesaj de ieșire 5 • aplicația 100		210
da — nodul la distanță	local	la distanță	• recepție mesaje (zece) pentru stocare disc 50 • accese la disc (zece) 50 • trimite mesaje (zece) cu înregistrări de pe disc 50		150

În situația că datele se iau de pe un sistem aflat la mare distanță, cererile și răspunsurile se introduc în mesaje care străbat rețeaua pînă la/de la calculatorul de destinație; efectul se traduce prin instrucțiunile ce trebuie executate suplimentar pentru partea de comunicație.

Partea a doua a tabelului indică numărul de instrucțiuni care se execută în nodul local și în cel aflat la distanță cînd execuția aplicației are loc în nodul local, iar datele sînt plasate în nodul aflat la distanță. Concluzia care se trage de aici este că plasarea execuției aplicației și a datelor asociate trebuie să se facă în același nod din rețeaua de calculatoare atunci cînd se dorește obținerea unui cost redus (acesta exprimat în instrucțiuni adiționale ce trebuie executate). Desigur vor fi cazuri cînd nu se poate ține seama de principiul enunțat mai sus. Pentru aceste cazuri costul va fi cu atît mai mare, cu cît aria geo-

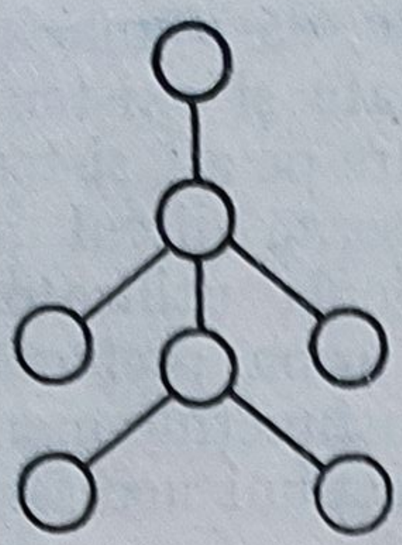


Fig. 4.8. Rețeaua funcțională pentru exemplu considerat:
○ unitate componentă a sistemului-obiect;
— legături de dependență care indică circulația informațiilor între entități.

grafică de întindere a rețelei de calculatoare va fi mai mare. La proiectarea rețelei de comunicații principiul general care trebuie urmat prevede localizarea execuției programelor și datelor astfel încît întîrzierile în rețea și costurile de comunicație să fie minime. Rețeaua organizațională pentru cazul prezentat în paragraful 4.2.3 este centralizată (pe tipul schemei din fig. 4.4—a.1). Pornind de la aceasta, rețeaua funcțională care definește repartiția datelor și a prelucrărilor se presupune de forma celei din figura 4.8. Datele sînt prelucrate și administrate cu ajutorul minicalculatoarelor plasate în diferite unități economice; pentru comunicațiile între mijloacele informatice se vor utiliza legături specializate.

Se propune precizarea mijloacelor informatice și a mijloacelor de telecomunicații pentru acest exemplu, definind astfel o rețea organică. Modul de plasare a execuției programelor, precum și a locului unde se memorează datele asociate va influența structura finală a configurației rețelei de calculatoare. Se introduc termenul flux-superior, care referă o prelucrare apropiată de calculatorul central (și depărtată de utilizator) și termenul flux-inferior, care semnifică o prelucrare apropiată de utilizator (și depărtată de calculatorul central). Conceptul flux-superior—flux-inferior, schițat în figura 4.9, va fi folosit în definitivarea configurației rețelei de calculatoare.

În nodurile rețelei (deci la dispeceratul național — DN — și în dispeceratele regionale — DR —) se plasează minicalculatoare, structura rețelei începînd astfel să se profileze (fig. 4.10). În punctele de dispecerat regional de întreprindere (DRI) se plasează stații de lucru (bazate, de exemplu, pe calculatoare personale compatibile IBM) care pot găzdui datele particulare, precum și execuția programelor asociate cu aceste date (în măsura în care capacitatea de calcul și memorare satisface cerințele). Pentru nivelul DRI

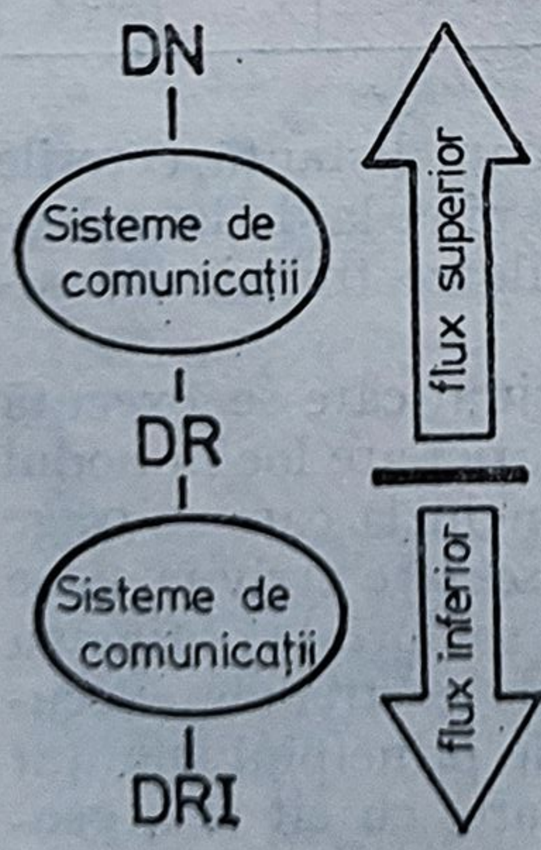


Fig. 4.9. Conceptul flux-inferior — flux-superior; *DN* — Dispecerat Național; *DR* — Dispecerat Regional; *DRI* — Dispecerat Regional de Întreprindere.

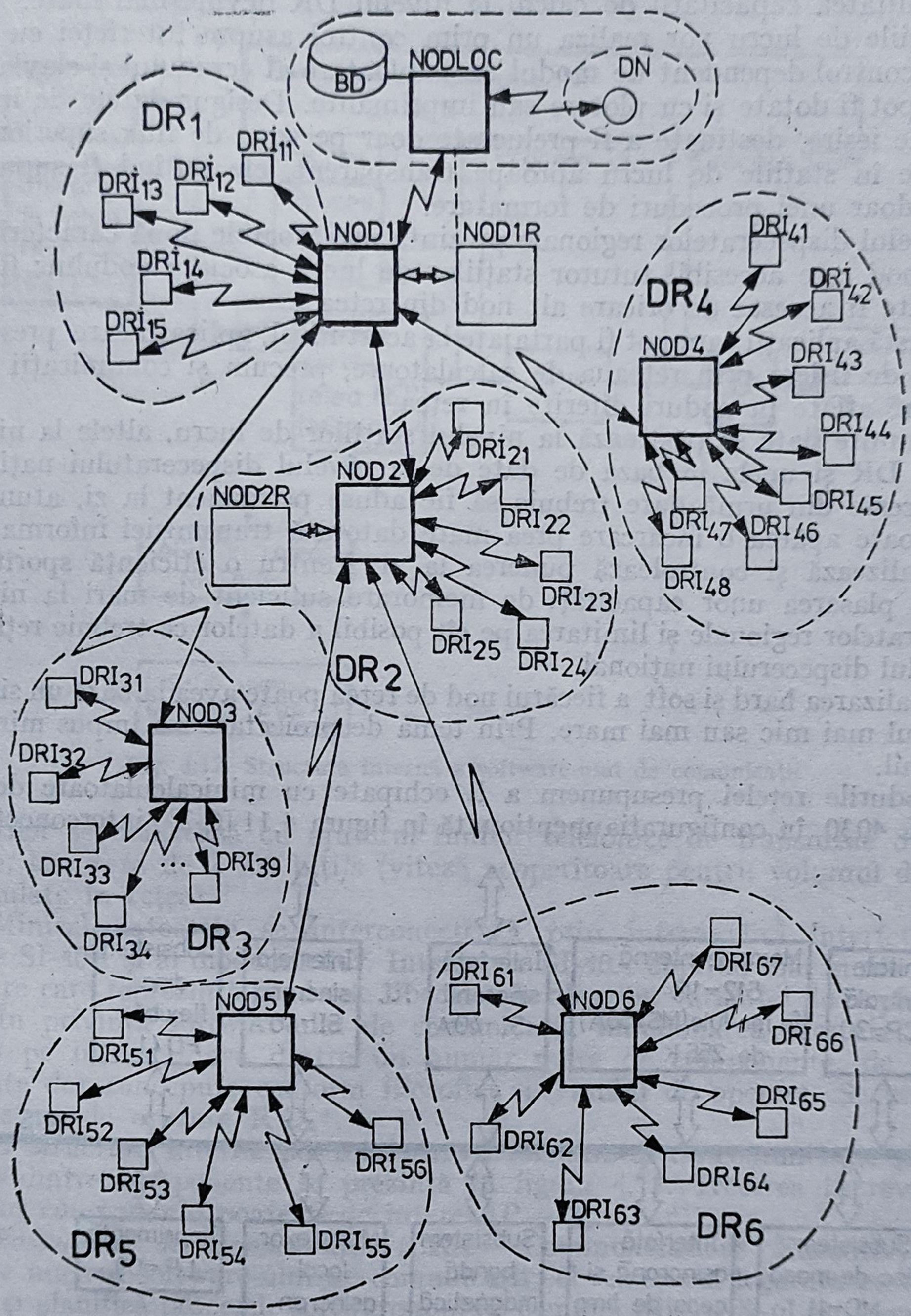


Fig. 4.10. Configurația rețelei pentru dispecerizarea gazelor naturale:

□ minicalculator (ex.: CORAL 4030);
□ stație de lucru (calculator personal);
○ display (DAF 2020);
DN Dispecer Național;
DR_n Dispecer Regional n;
DRI_{n1} Dispecer Regional de Întreprindere;
BD Baza de Date.

atît timpul de răspuns, cît și costurile pentru comunicații sînt reduse, iar disponibilitatea capacității de calcul la nivelul DR devine mai mare.

Stațiile de lucru vor realiza un prim control asupra interfeței cu utilizatorul, control dependent de modul de exploatare al ecranului și claviaturii. Stațiile pot fi dotate și cu plotere sau imprimante. Desigur datele de intrare și cele de ieșire, destinate a fi prelucrate doar pe zona de flux superior, vor fi privite în stațiile de lucru aproape transparent, ele putînd fi supuse în acestea doar unei proceduri de formatare.

Nivelul dispeceratelor regionale prezintă următoarele două caracteristici: fiecare nod este accesibil tuturor stațiilor de lucru asociate nodului; fiecare nod poate fi accesat de oricare alt nod din rețea.

Există aplicații care pot fi partajate la acest nivel, aplicații care presupun transfer de fișiere prin rețeaua de calculatoare, precum și comunicații între programe aflate în noduri diferite în rețea.

Anumite date se păstrează la nivelul stațiilor de lucru, altele la nivelul fiecărui DR și unele în baza de date de la nivelul dispeceratului național. Dacă aceste din urmă date trebuie să fie aduse permanent la zi, atunci în rețea poate apărea o încărcare prea mare datorată transmisiei informațiilor care realizează și controlează punerea la zi. Pentru o eficiență sporită se impune plasarea unor capacități de memorare suficient de mari la nivelul dispeceratelor regionale și limitarea pe cît posibil a datelor ce trebuie reținute la nivelul dispecerului național.

Realizarea hard și soft a fiecărui nod de rețea poate avea la bază un sistem de calcul mai mic sau mai mare. Prin tema de proiectare s-a impus minicalculatorul.

Nodurile rețelei presupunem a fi echipate cu minicalculatoare de tip CORAL 4030, în configurația menționată în figura 4.11 [C-2]; interconectarea

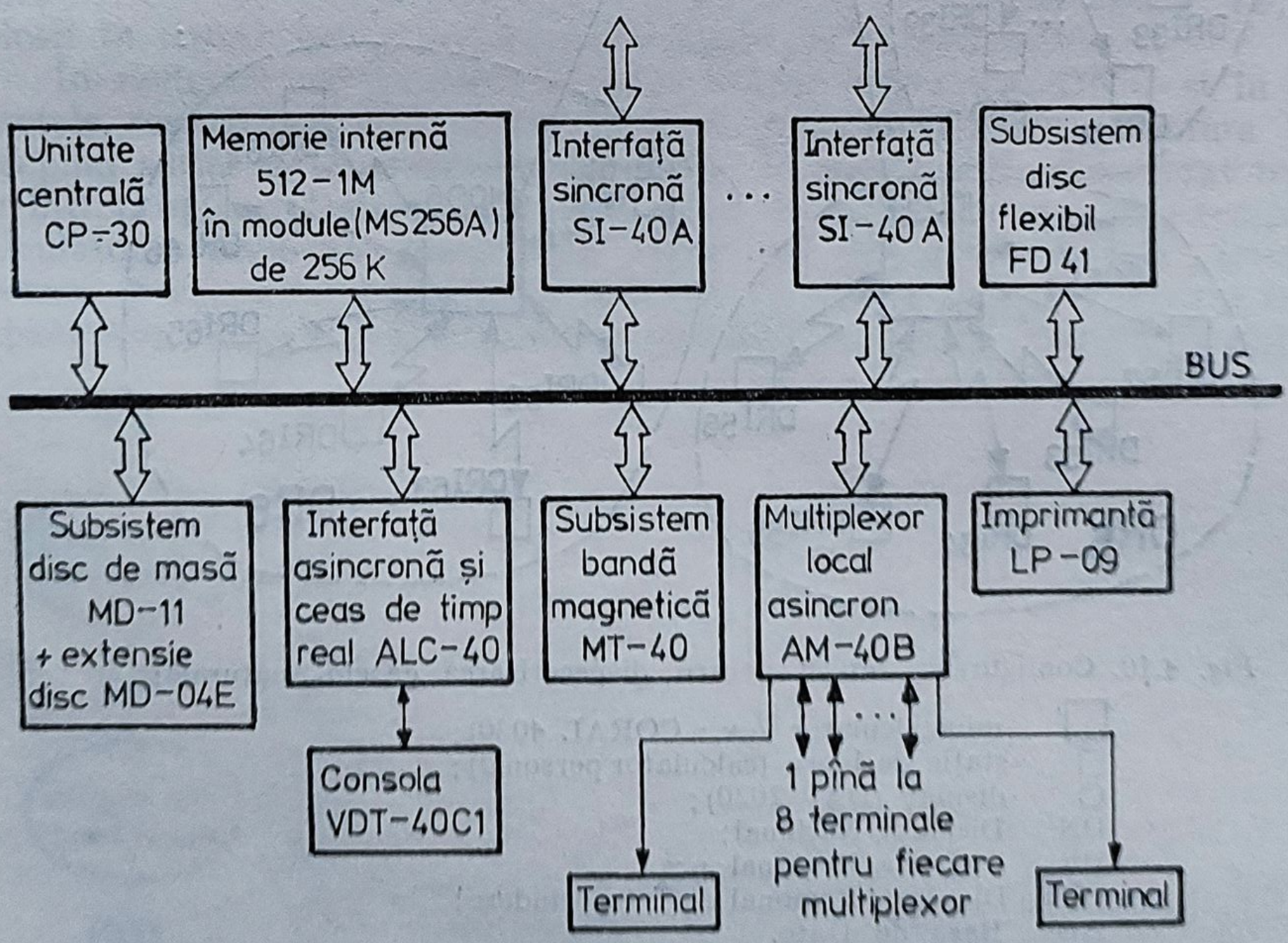


Fig. 4.11. Configurația minicalculatorului CORAL 4030 din nodul rețelei

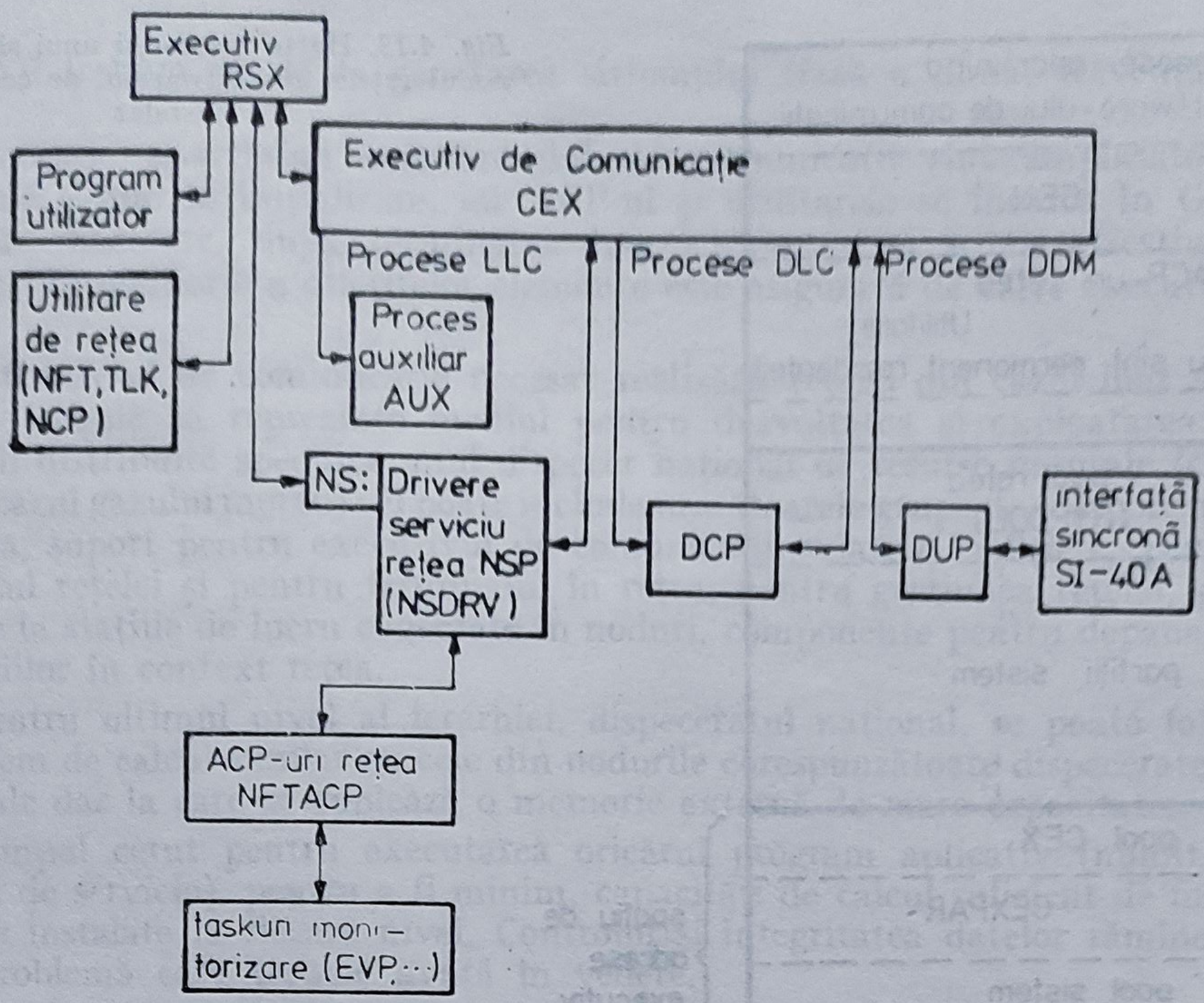


Fig. 4.12. Structura internă a software-ului de comunicații

nodurilor se realizează cu ajutorul liniilor telefonice de transmisie date cu 4 fire, la viteza de 2 400 biți/s (viteză acoperitoare pentru volumul de date vehiculate în rețea).

Minicalculatoarele se interconectează prin intermediul interfeței sincrone SI-40A și al modemurilor. Interfețele SI-40A cuprind mai multe comutatoare care reprezintă grade de libertate în alegerea variantei de exploatare.

În privința software-ului de comunicații, funcționarea acestuia se bazează pe interacțiunea dintre un număr mare de componente, de naturi diferite dar concepute conform filosofiei sistemului de operare. Se propune ca sistem de operare RSX*.

O structură internă posibilă pentru software-ul de comunicație și legăturile dintre componente se prezintă în figura 4.12. Trecerea în revistă a acestor componente poate fi de interes [P — 1]:

Executivul de Comunicație (CEX — Communications Executiv) constituie nucleul software-ului de comunicații; el realizează legătura cu executivul și planifică procesele software-ului de comunicații. CEX-ul este conceput ca o extensie a executivului, fiind rezident în memorie în spațiul de adresă al executivului (în partiția CEXPAR).

Pool-urile constituie zone de memorie dinamică și pot fi, funcție de utilizare, de

- sistem, caz în care sînt folosite pentru comunicația între taskuri și executiv; în acestea se alocă blocul de QIO ce reprezintă suportul soft al cererilor către software-ul de comunicații din partea utilizatorilor;

* elaborat de firma DEC.

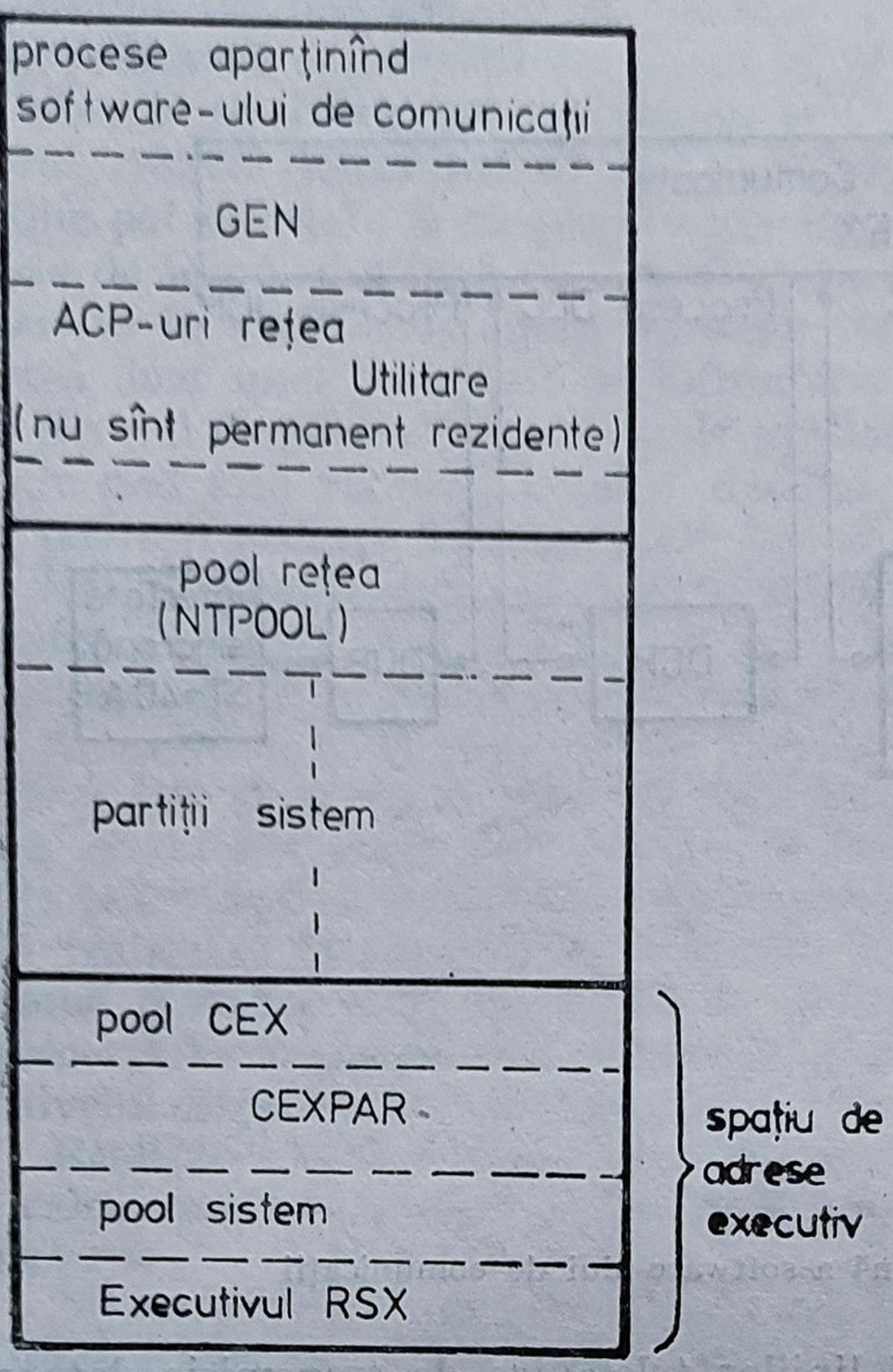


Fig. 4.13. Harta memoriei unui sistem din nodul rețelei cu software-ul de comunicații instalat

- CEX, caz în care sînt folosite de CEX în comunicația cu executivul;
- rețea, caz în care sînt folosite ca spațiu pentru buffere, pentru mesaje.

Drivere de serviciu rețea. Cererile utilizatorilor către software-ul de comunicații au ca suport directive executiv QIO către periferice fictive: NS:, NX:.... . Principalul driver îl constituie NSDRV care preia aceste cereri de la executiv și le transmite CEX-ului; în ierarhia software-ului de comunicații este un proces de tip LLC.

ACP-urile de rețea reprezintă task-uri privilegiate ce asistă driverele de rețea în activitatea lor. Driverul este o componentă de dimensiuni reduse (maximum 4 k cuvinte), rezidentă în memorie, ca urmare va executa doar sarcinile critice în timp, restul sarcinilor revenind ACP-ului. Lui NSDRV îi corespunde NETACP. Software-ul de comunicații vede perechea NSDRV—NETACP ca un tot, și anume procesul NSP.

Procesele reprezintă componente cu rolul de a rezolva sarcini în timp critic; acestea sînt similare unui driver RSX dar nu sînt recunoscute ca atare (cu excepția NSP-ului). Procesele sînt de trei tipuri LLC, DLC și DDM; rezidente în memorie ele stau în partiția GEN.

Utilitarele de rețea reprezintă task-uri RSX ce comunică cu nucleul CEX prin același mecanism de QIO și care au diverse funcții de control rețea, monitorizare, transfer.

În figura 4.13 este reprezentată harta memoriei sistemului de operare dintr-un nod al rețelei cu software-ul de comunicații instalat. Partiția

CEXPAR trebuie creată la generarea sistemului (faza a doua, secvență de VMR).

Procesele aparținînd software-ului de comunicații sînt amplasate în vîrful GEN-ului la inițializare, iar ACP-ul și utilitarele se încarcă în GEN cînd sînt necesare, după terminarea funcției putînd fi scoase. Gestiunea prezenței în memorie a diferitelor elemente este asigurată de către executivul RSX.

Software-ul de comunicație necesar realizării rețelei din cazul luat spre studiu, trebuie să reprezinte mediul pentru dezvoltarea și exploatarea de aplicații distribuite specifice unui dispecer național de resurse naturale (cum este și cazul gazului metan); el poate include următoarele grupe de componente: de bază, suport pentru executivul de comunicații, relative la fișiere, pentru controlul rețelei și pentru terminalul în rețea, pentru gestiunea rețelei, privitoare la stațiile de lucru conectate în noduri, componente pentru depanarea aplicațiilor în context rețea.

Pentru ultimul nivel al ierarhiei, dispeceratul național, se poate folosi un sistem de calcul similar cu cele din nodurile corespunzătoare dispeceratelor regionale dar la care se cuplează o memorie externă de mare capacitate.

Timpul cerut pentru executarea oricărui program aplicativ (numit și timpul de serviciu), pentru a fi minim, capacități de calcul suficient de mari trebuie instalate la fiecare nivel. Controlul și integritatea datelor rămîne o altă problemă care trebuie avută în vedere.

Ca o concluzie finală se estimează că pentru următorii ani sisteme de calcul diferite vor continua să fie utilizate împreună în regim rețea, fiecare tip prezentînd avantaje specifice pentru utilizare. Parametrul fundamental în dezvoltarea unor asemenea sisteme complexe devine portabilitatea.

4.3.3. PROBLEME ÎN PROIECTAREA REȚELELOR INTEGRATE

Astăzi se folosesc două rețele de telecomunicații independente, una dedicată vocii (și utilizată în aplicații ce implică teletypuri și facsimiluri) și alta dedicată comunicațiilor de date, fiecare dispunînd de interfețe, mecanisme de transport și comutare specifice.

Sistemele teleinformatice cer tot mai frecvent ca serviciile oferite de cele două rețele să fie combinate și oferite de o rețea unică și aceasta în condițiile extinderii gamei de echipamente, al scăderii costului comunicațiilor, al obținerii unei înalte calități pentru serviciile care trebuie să fie tot mai diversificate. Cu această soluție de combinare a celor două rețele de telecomunicații se va merge pînă după anii 1990. Direcția de dezvoltare însă o reprezintă rețeaua digitală cu servicii integrate (ISDN — Integrated Services Digital Network) care între altele va prezenta o interfață unică pentru acces la servicii.

Principalele probleme pe care le ridică partea de comunicații între calculatoare într-un sistem teleinformatic, sînt cele legate de transport și comutare.

Comutarea de circuite convențională se utilizează pentru traficul vocal iar pentru traficul de date se utilizează comutarea de pachete care poate sau nu să fie rapidă. Cele două tehnici de comutare se pot realiza pe arhitecturi de rețele diferite sau într-o manieră modernă în cadrul aceleiași rețele, cu

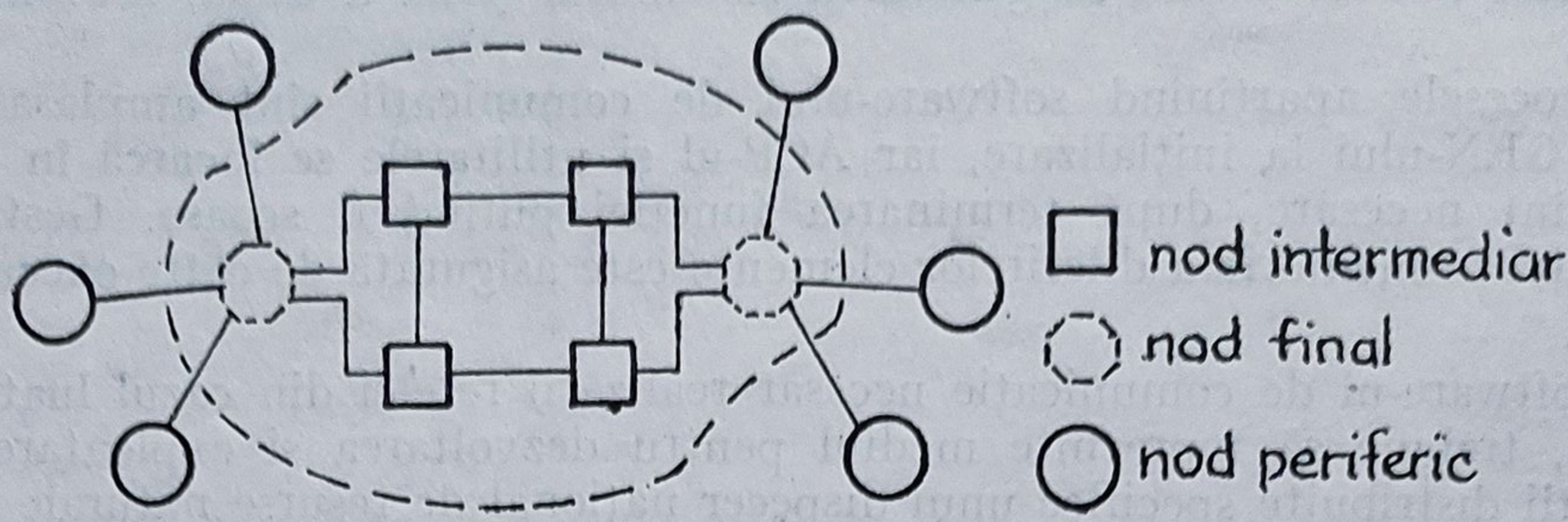


Fig. 4.14. Model pentru o rețea de calculatoare larg răspîndită geografic

toate că procesarea vocii și a datelor se face diferit (de exemplu în ceea ce privește bufferarea, blocarea etc.).

Proiectantul de sisteme teleinformatice este confruntat cu alegerea formei de comutare; să fie aceasta comutarea de pachete simplă sau o combinație a comutării de pachete cu comutarea de circuite?

În ceea ce privește comutarea rapidă de pachete aceasta menține caracteristicile comutării convenționale de pachete, cu accent însă pe o comutare mai rapidă și pe o procesare minimă în nodul intermediar pentru controlul fluxului, al rutării și al refacerii în caz de erori de transmisie.

Se poate utiliza și o comutare hibridă în care comutarea convențională de circuite se împletește cu comutarea rapidă de pachete.

Un model pentru o rețea de calculatoare larg răspîndită geografic care reprezintă suportul pentru un sistem teleinformatic, poate fi cel din figura 4.14.

Modelul cuprinde două părți distincte: una, cea inclusă în interiorul liniei întrerupte (dedicată comunicațiilor) iar cea de-a doua alcătuită din mulțimea nodurilor periferice.

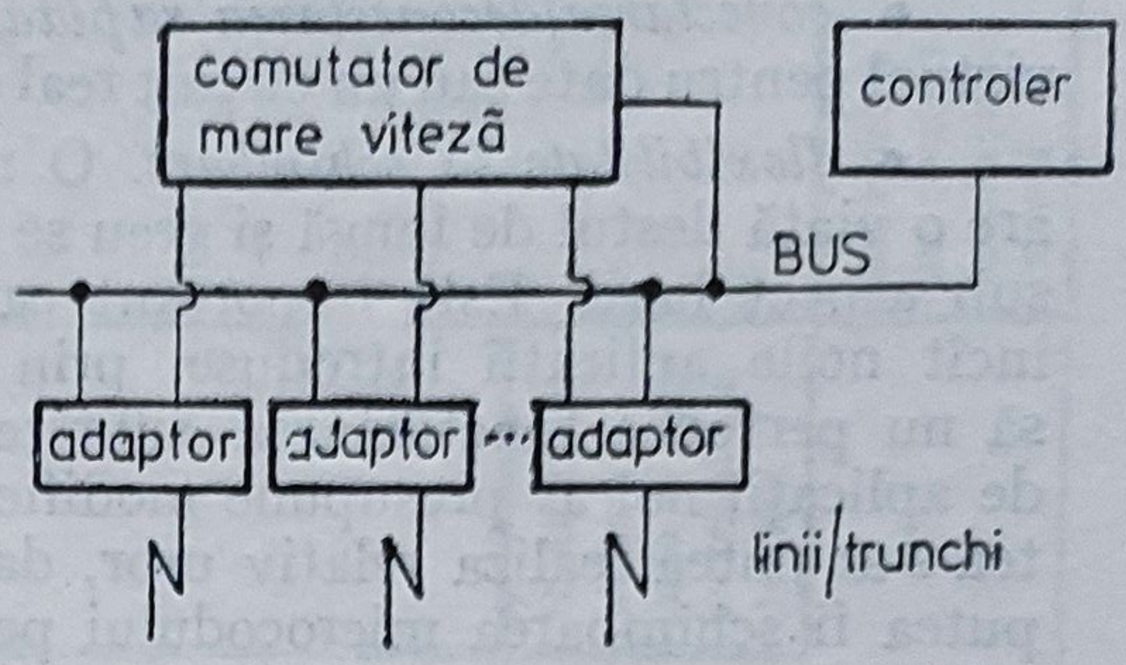
Nodurile periferice sînt formate din baza instalată de calculatoare și terminale, controloare, procesoare etc. ale utilizatorului, conectate la subrețeaua de comunicații prin linii de joasă viteză; ele nu contribuie la procesul de rutare, dar sînt puternic implicate în execuția aplicațiilor.

Subrețeaua de comunicații cuprinde nodurile intermediare conectate între ele prin trunchiuri de mare viteză (pînă la zece pe nod). Pentru un trunchi individual, partiționarea capacității între traficul de pachete și cel vocal este realizată prin interacțiunea controloarelor aflate la capetele trunchiului. Numărul nodurilor intermediare este cuprins între cîteva bucăți și zeci sau sute; aceste noduri contribuie din plin la procesul de rutare. Tot din subrețeaua de comunicații fac parte și nodurile finale care contribuie și ele la rutare; acestea suportă atît interfețele pentru trunchi cît și cele pentru linie. Fie că este vorba de un nod intermediar sau de un nod final, atît în cazul comutării rapide de pachete cît și în cazul comutării hibride, o arhitectură posibilă de nod este cea prezentată în figura 4.15.

Comutatoarele de mare viteză sînt în număr de unu pentru comutarea rapidă de pachete și două pentru cazul comutării hibride. Adaptoarele pot realiza atît o multiplexare cît și o comutare pentru viteze mici.

Aplicațiile se execută în nodurile periferice și eventual în nodurile terminale.

Fig. 4.15. Arhitectura nodului intermediar sau final



Viteza de transmisie pe trunchi merge azi pînă la 1,5 Mbiți, iar în cazul folosirii sateliților la multimegabiți.

Pentru cazul cînd în sistemul teleinformatic proiectat nu se folosește o rețea publică de date, în proiectarea în special a subrețelei de comunicații trebuie ținut seama de o serie de cerințe formulate față de aceasta, între ele numărîndu-se:

- *disponibilitatea*, care prezintă două aspecte, primul legat de satisfacerea cît mai rapidă a unei cereri de conexiune între două obiecte logice, unul aflat în nodul periferic sursă celălalt în nodul periferic destinație și al doilea aspect legat de menținerea legăturii stabilite. Pentru primul aspect parametrul admis va trebui să fie sub 1%, adică dintr-o sută de cereri de conexiune doar una are voie să nu fie satisfăcută.

- *integritatea mesajelor vocale și de date*. Mesajele pot să se piardă în rețea din diverse motive ca, de exemplu: erori de transmisie pe trunchi, depășirea capacității bufferelor de acumulare din noduri, utilizarea unei metode slabe de control a congestiei etc. Dacă întreruperea semnalului vocal poate fi tolerată pentru scurte perioade de timp (sub 15 ms) pentru evitarea pierderii mesajelor de date se impun tehnici sofisticate de detectare și retransmisie, și aceasta mai cu seamă între nodurile terminale.

- *întîrzierea capăt-la-capăt*. Întîrzierea mesajelor în drumul lor dintre două noduri terminale este determinată și de suportul fizic folosit în comunicații. Astfel legăturile realizate prin sateliți sau în sistemul clasic de joasă viteză sau într-o rețea cu comutare de pachete unde nodurile sînt legate prin trunchiuri de viteze mici introduc mari întîrzieri în transportul mesajelor. În cazul mesajelor vocale apar efecte neplăcute de ecou pentru care însă astăzi există tehnici moderne de îndepărtare. Pentru circuitele vocale terestre se recomandă (în conformitate cu CCITT) întîrzierea de propagare maximă pentru o direcție să nu depășească 80 ms. Studii recente au arătat că în cazul mesajelor de date, în multe aplicații performanțele sistemelor teleinformatice sînt degradate de timpii de răspuns prea mari. De aceea ca parametru de proiectare se recomandă valoarea de 250 ms ca limită maximă admisibilă pentru întîrziere pe o direcție într-un schimb interactiv de date.

- *complexitatea implementării*. Pentru o rețea deja existentă care ar urma să fie folosită într-un sistem teleinformatic, înțelegerea complexității rețelei și cunoașterea modului de administrare a acesteia devin importante.

- *utilizarea trunchiului*. Aceasta se face de obicei mult sub valoarea permisă.

- *viteza pentru liniile abonaților*.

• *conectarea/deconectarea rapidă.* Viteza cu care se stabilește un circuit virtual pentru date sau un circuit real pentru voce trebuie să fie cît mai mare.

• *flexibilitate la schimbare.* O rețea de telecomunicații odată instalată are o viață destul de lungă și greu se dezvoltă atît sub aspect soft cît mai ales sub aspect hard. Este important ca proiectarea să se facă de așa manieră încît noile aplicații introduse, prin traficul suplimentar pe care-l implică să nu perturbe funcționarea subrețelei de comunicații. Dacă introducerea de aplicații noi ar presupune modificări soft în nodurile de comunicații acestea s-ar putea realiza relativ ușor, dar o modificare de tip hardware (cum ar putea fi schimbarea microcodului pentru o modificare în nivelul fizic al rețelei) este mult mai dificil de realizat.

4.4. PROIECTARE DE DETALIU

Tehnologia de realizare a sistemelor teleinformatice, după ce parcurge etapele de elaborare a temei de realizare și proiectare de ansamblu continuă cu proiectarea de detaliu.

4.4.1. TEHNOLOGIA ETAPEI

Proiectarea de detaliu a unui sistem teleinformatic își are obiectivele ei concrete, și anume:

• analiza și specificarea cerințelor de detaliu la nivel de componentă funcțională/sistem de administrare a rețelei de calculatoare (SARC) și a bazei/bazelor de date (SABD);

• proiectarea arhitecturii și proiectarea tehnică a componentei funcționale/sistemului de administrare a rețelei de calculatoare și a bazei/bazelor de date;

• planificarea realizării și punerii în funcțiune a componentei funcționale;

• planificarea și pregătirea testării.

Proiectarea de detaliu a sistemelor teleinformatice cuprinde la rîndul ei un șir de activități — corespunzătoare pașilor ce trebuie parcurși — cum sînt: tehnice de realizare, de planificare a procesului de realizare, de pregătire a testării, de autocontrol (pentru asigurarea calității), de realizare a formei finale a proiectului de detaliu, de control (normativ și metodologic) efectuat la sfîrșitul etapei.

Legăturile dintre intrări — activități — ieșiri în proiectarea de detaliu sînt date în tabelul 4.3.

În etapa proiectării de detaliu activitățile tehnice de realizare includ o proiectare logică și una tehnică, care vizează (fiecare din ele) componentele funcționale, sistemul de administrare a rețelei de calculatoare și sistemul de administrare a bazei de date (SABD).

Faza proiectării logice a componentei funcționale prevede: analiza și specificarea cerințelor de detaliu, proiectarea arhitecturii acesteia (structuri funcționale și interfețe), definirea componentelor funcționale la nivel inferior

TABELUL 4.3 Activităţi (asociate cu intrări şi ieşiri) în proiectarea de detaliu

Intrări \ Ieşiri	Proiect logic/specificaţia de definire a componentei funcţionale	Proiect logic/specificaţia de definire a SARC	Activităţi de control la sfîrşitul etapei	Raport de evaluare a etapei
Proiect de ansamblu	(proiectare logică)			
Proiect logic/specificaţii de definire componentă, SARC, SABD	Activităţi tehnice de realizare		Activităţi de planificare a procesului de realizare	
Planul de testare de integrare componentă funcţională/SARC/SABD				
Produse intermediare				
Produse intermediare				
Proiect de detaliu al componentei funcţionale /SARC/SABD				

Intrări \ Ieşiri	Proiect logic/specificaţia de definire a componentei funcţionale	Proiect logic/specificaţia de definire a SARC	Proiect logic/specificaţia de definire a SABD	Proiect tehnic/specificaţia de realizare a componentei funcţionale	Proiect tehnic/specificaţia de realizare a SARC	Proiect tehnic/specificaţia de realizare a SABD	Planul realizării şi punerii în funcţiune a componentei funcţionale/SARC/SABD	Planuri de testare individuală (a programelor)	Planul de testare şi integrare componentă funcţională/ SARC/SABD	Specificaţia de testare pentru integrare componentă funcţională/SARC/SABD	Produse intermediare (validate intern)	Proiect de detaliu al componentei funcţionale/SARC/SABD
(proiectare logică)												
Activităţi tehnice de realizare												
Activităţi de planificare a procesului de realizare												
Activităţi de pregătire a testării												
Activităţi de autocontrol												
Realizarea formei finale a proiectului de detaliu												
Activităţi de control la sfîrşitul etapei												

precum și proiectarea structurii logice și a soluției de gestionare a colecției de date proprie componentei funcționale.

Proiectarea logică a componentei de administrare a rețelei de calculatoare începe cu analiza și specificarea cerințelor de detaliu ale SARC; se identifică poziția, fluxul, debitul datelor și se definesc aspectele dimensionale. Proiectarea logică de detaliu a SARC impune analiza cerințelor de date ale subsistemelor/componentelor funcționale, validarea modelului conceptual pentru programele de comunicație, proiectarea structurii și conținutului programelor de comunicație din toate tipurile de noduri ale rețelei, definirea funcțiilor SARC, definirea interfețelor cu celelalte componente funcționale/sisteme și cu utilizatorul, specificarea cerințelor și restricțiilor tehnice de realizare a SARC.

Proiectarea logică și tehnică a SABD precum și proiectarea tehnică a componentei funcționale se fac conform indicațiilor metodologice precizate în [T-2].

Proiectarea tehnică a SARC demarează cu proiectarea structurii fizice a acesteia (se identifică programele de comunicație, relațiile statice și dinamice dintre ele și dintre ele și unele din programele de aplicație precum și procedurile automate/manuale).

Se proiectează interfața programelor de comunicații cu programele de aplicație și se specifică criteriile de validare; se construiesc procedurile de execuție din cadrul SARC și se proiectează fluxul de control la nivelul SARC.

În etapa proiectării de detaliu, tehnologia de realizare a STI folosește metode, tehnici și instrumente variate, dintre care menționăm: analiza celulară, analiza diagnostic, modelarea de sistem orientată pe obiecte, tabele de decizie, proiectarea structurată, proiectarea programelor de comunicații orientată pe obiecte, facilități SGBD, diagrame flux de date, normative de exploatare a sistemelor de comunicații oferite de administrația PTT, recomandări CCITT, ș.a.

Documentațiile care înglobează rezultatele activității de proiectare de detaliu și constituie forme finale ale produsului etapei sînt reprezentate de proiectul logic/specificația de definire și proiectul tehnic/specificația de realizare, a căror conținut se precizează în [T-2].

4.4.2. STUDIU DE CAZ

În sistemul de dispecerizare la nivel național a gazelor naturale, a cărui ierarhie stabilită în sistemul-obiect a fost prezentată în fig. 4.3, pentru urmărirea și calculul debitelor de gaze se pot furniza informații consultabile la orice moment de timp. Nodurile rețelei de minicalculatoare din cadrul configurației pentru dispecerizarea gazelor naturale (prezentată în fig. 4.10) găzduiesc și execuția calculelor referitoare la debitele de gaze, aceasta reprezentînd una din funcțiunile sistemului teleinformatic al cazului luat în studiu [L-4].

Realizarea tuturor funcțiunilor sistemului de dispecerizare a gazelor naturale se regăsește în soluțiile tehnice de structurare și organizare a datelor și prelucrărilor acestora, precum și în satisfacerea informațională prin ieșiri; acestea se grupează după funcțiunile de bază, un tip de ieșiri (liste, grafice) putîndu-se referi la dispecerizarea generală (codificată de noi DG).

Variantele acestor ieșiri se propun funcție de gradul de detaliu al unuia din parametrii urmăriți (presiune tehnologică, temperatură, debit instantaneu) în cadrul rețelei de transport; exemplificarea, prezentată în tabelul 4.4 se referă la partea de debite din dispecerizarea generală.

TABELUL 4.4 Variante de ieșiri referitoare la debite în dispecerizarea generală

Nr. crt.	Ieșire		Variante		Suport		Utilizatori
	Cod	Denumire	Cod	Caracteristici	Term.	Imprim.	
1	DG.01	Debite instantanee la data ... ora ... (liste)	V1 V2 V3 V4 V5	— pe întreg teritoriul — pe sisteme transport — pe ramuri — pe conducte — pe zone geografice	* * * * *	* * * * *	dispeceri din dispeceratul regional
2	DG.02	Debite instantanee la data ... ora ... (grafice)	V1 V2 V3 V4 V5	— pe întreg teritoriul — pe sisteme transport — pe ramuri — pe conducte — pe zone geografice	* * * * *	* * * * *	dispeceri din dispeceratul regional
3	DG.03	Evoluție debite instantanee în ultimele 6/24 ore (liste)	V1 V2 V3 V4	— pe întreg teritoriul — pe sisteme transport — pe ramuri — pe zone geografice		* * * *	dispeceri din dispeceratul național
4	DG.04	PLC-uri în care debitele instantanee la data ... ora ... sînt ... (liste)	V1 V2	— val. minimă — val. maximă	* *	* *	dispeceri din dispeceratul național
5	DG.05	Debite de gaze livrate în ziua ... ora ... (liste)	V1 V2 V3 V4	— pe întreg teritoriul — pe sisteme transport — pe ramuri — pe zone geografice	* * * *	* * * *	dispeceri din dispeceratul regional și național

Pentru a răspunde cerințelor sistemului teleinformatic datele s-au organizat folosind principiile bazelor de date. Există o bază de date centrală (BDC) situată fizic pe discurile de masă din Centrul de prelucrare centralizată a datelor și baze de date regionale (BDR) situate pe discurile de masă ale calculatoarelor din nodurile rețelei (DR).

Bazele de date cuprind zeci de fișiere și sînt exploatate de programe scrise în unul din limbajele FORTRAN 77, COBOL sau MACRO.

Pentru realizarea sincronizării evenimentelor în cadrul bazelor de date (răspîndite geografic) și între ele s-a impus crearea unui fișier de evenimente în fiecare bază de date. În bazele de date regionale aceste fișiere se numesc EVEL.DAT iar în baza de date centrală EVEC.DAT.

În organizarea și structurarea prelucrării datelor s-au avut în vedere o serie de principii prezentate și în [L-4], în conformitate cu care prelucrările s-au organizat și structurat pe module de prelucrare, proceduri de prelucrare și programe. Fiecare modul de prelucrare se compune din una sau mai multe proceduri de prelucrare. Pentru exemplificare se prezintă în figura 4.16 (a

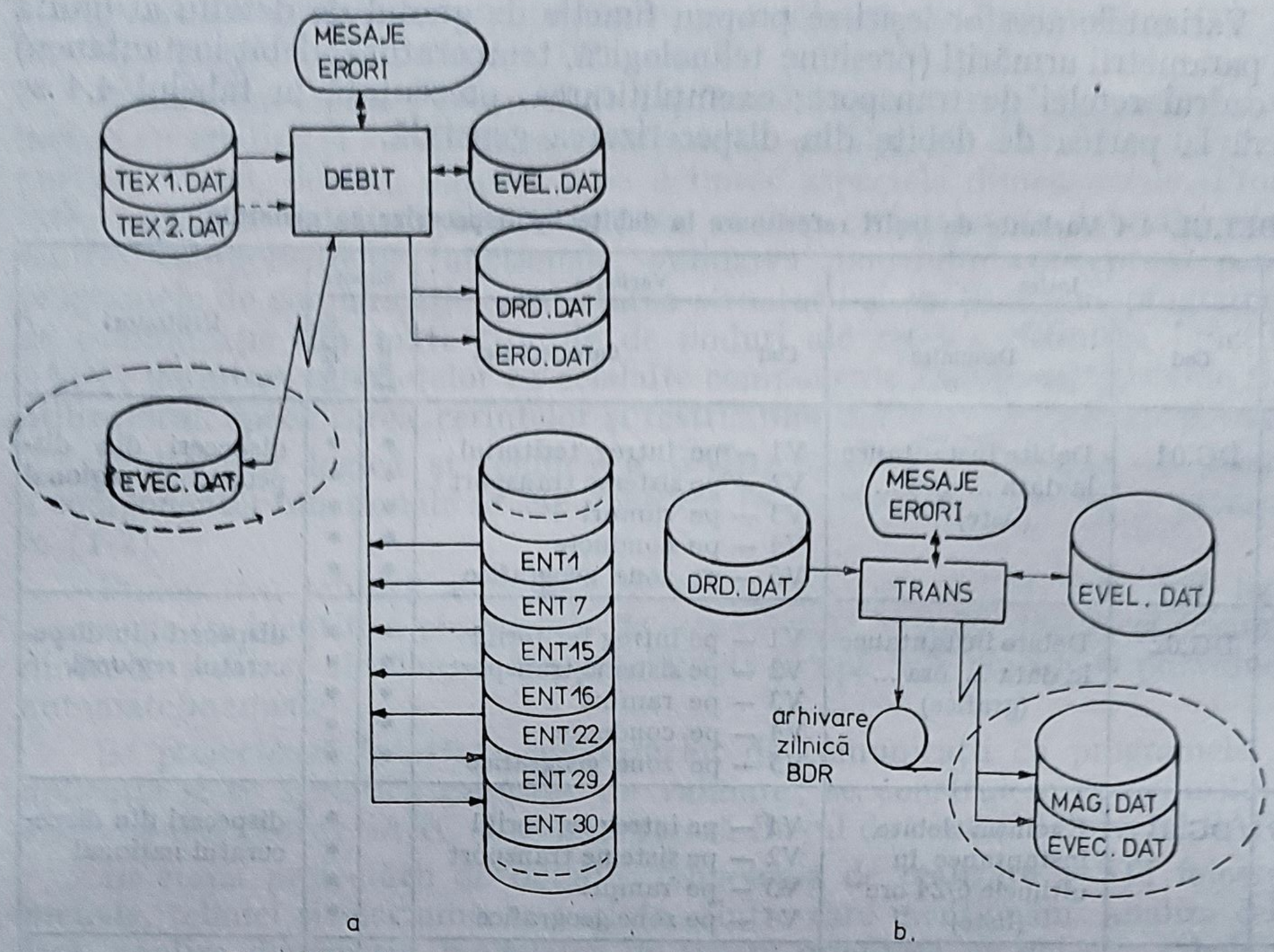


Fig. 4.16. Scheme de prelucrare într-un proiect de detaliu:
a. calculul debitului de gaze; *b.* transmiterea datelor la BDC și arhivare locală.

și b) schemele de prelucrare pentru calculul debitului de gaze și respectiv transmiterea datelor la BDC și arhivare locală ambele din cadrul modulului de actualizare BDR.

În procedura automată DEBIT, după actualizarea entităților BDR se calculează debitele de gaze aferente tuturor PLC-urilor ce au transmis date în fișierele TEX1.DAT și TEX2.DAT. Debitele calculate se adaugă în fișierul DRD.DAT, iar pe ecranul monitorului se semnalează mesaje de eroare și control din timpul calculului debitelor. Erorile datorate incompletitudinii parametrilor se scriu în fișierul ERO.DAT.

Comunicațiile prin rețeaua de calculatoare intervin în această procedură în actualizarea fișierului central de evenimente.

Procedura automată TRANS asigură transferul, sub controlul programului monitor local al fișierului de date local DRD.DAT prin rețeaua de calculatoare în fișierul MAG.DAT din nodul central. Comunicațiile între calculatoare intervin în procedură și la actualizarea fișierului central de evenimente.

4.5. ELABORARE DE PROGRAME; INTEGRARE ȘI TESTARE

În elaborarea programelor pentru un sistem teleinformatic, în integrarea și testarea lor, datorită faptului că sînt implicați un număr mare de proiectanți cu specialități nuanțate, există o serie de factori care joacă un rol deter-

minant: organizarea și structurarea activităților, asigurarea dezvoltării produselor intermediare și a unei calități superioare pentru acestea, instruirea personalului, metodele, tehnicile, instrumentele și normativele disponibile.

Programarea a evoluat de la o profesiune cu un puternic caracter individual către una ce presupune o activitate inginerească ordonată. În dezvoltarea programării un pas important s-a făcut în momentul în care oamenii au învățat cum și ce trebuie să facă și mai ales cum pot dirija și estima activitatea de programare într-un mod controlat.

În ultimul timp numărul programatorilor a crescut considerabil, de asemenea s-a dezvoltat nivelul abilităților de programare, s-au dezvoltat metodele și instrumentele folosite.

Pentru cei ce conduc activitatea de programare destinată unui sistem teleinformatic se cere ca aceștia să fi lucrat anterior în programare, să înțeleagă programarea, să aibă o instruire în direcția conducerii de proiecte, să cunoască un număr cît mai mare de produse program (atît cu caracter aplicativ cît și orientate pe comunicații de date) și să aibă noțiuni tehnice privind principiile de bază ale modului de realizare a comunicațiilor de date între calculatoarele dintr-o rețea.

Pentru a obține un software de comunicație și aplicativ lipsit de erori, există o serie de principii care pot guverna o activitate de dezvoltare de programe de calitate superioară, principii care pot fi urmate, ca de exemplu:

- implicarea profesioniștilor în procesul de programare;
- definirea riguroasă a activităților care se impun pentru elaborarea, integrarea și testarea programelor;
- culegerea de date statistice și analizarea acestora pentru identificarea din timp a eventualelor probleme ce pot să apară;
- controlul direct al procesului de dezvoltare de programe prin: controlul oricăror schimbări care apar pe parcurs, identificarea și luarea în considerare a oricăror îmbunătățiri necesare;
- instruirea profesională eficientă și corectă;
- utilizarea celor mai bune metode, tehnici, instrumente existente și eventual dezvoltarea unora care să corespundă noilor cerințe.

Pentru desfășurarea unei munci competente trebuie realizat un control permanent al întregului context în care se desfășoară dezvoltarea de programe în scopul de-a nu risipi ingeniozitatea, priceperea și energia specialiștilor pentru rezolvarea unor probleme deja rezolvate.

4.5.1. ARHITECTURA DEZVOLTĂRII DE PROGRAME PENTRU UN STI

În ultimul timp au fost create diverse și multiple produse software atît pentru calculatoare lucrînd independent cît și pentru cele interconectate în rețea.

Se consideră că dezvoltarea de programe cuprinde elaborarea acestora, integrarea și testarea lor.

O arhitectură a dezvoltării de programe pentru un STI poate contribui la asigurarea unui mod unitar de dezvoltare și de utilizare a software-ului deja existent, constituind totodată o bază teoretică solidă, un cadru conceptual unificat care poate conduce la o activitate de programare coerentă [I-2].

Stabilirea unei arhitecturi ar putea avea ca obiective: determinarea celui mai bun software existent, punerea de acord a unui model privind evoluția software-ului (atît de aplicație cît și de comunicație) în toate tipurile de noduri din rețeaua suport a sistemului teleinformatic, obținerea unui model care să permită maparea rapidă pe un alt nod din rețea. Cele prezentate mai sus au semnificație în condițiile în care majoritatea produselor software care trebuie să fie exploatate împreună într-un STI au fost scrise în coduri diferite, unele din ele avînd aceleași obiective au fost dezvoltate concurent, nivelele de punere la zi diferă, noile produse dezvoltate în aceleași laboratoare au beneficiat de forme noi și variate de programare. Deci este evident că la nivele superioare procesul de dezvoltare a software-ului trebuie să conducă la un software omogen în condițiile în care în multe din nodurile rețelei de calculatoare, dacă ar fi să exemplificăm doar cu sistemul de operare (prin urmare la nivele scăzute) există mari diferențe.

În proiectele mari care includ și o rețea de calculatoare se poate agrea utilizarea unei arhitecturi a dezvoltării de programe pentru a obține o structură comună a software-ului sistemelor din rețea, pentru ca evoluția pe scară largă a întregului sistem teleinformatic să culmine cu o a „*n*-a arhitectură" (finală). Obiectivul unei arhitecturi finale ar putea fi obținerea de zero defecte, fără vreo abatere de la specificații.

Prezentăm în continuare cîteva sugestii privind crearea unei arhitecturi a dezvoltării de programe pentru realizarea unui sistem teleinformatic.

La baza creării arhitecturii dezvoltării de programe, atunci cînd se realizează un sistem teleinformatic, stă ideea „definirii a ceea ce trebuie făcut" în direcția software-ului.

Arhitectura se definește astfel încît să țină seama de următoarele principii:

• dezvoltarea de programe trebuie să fie supravegheată activ, continuu și consistent pentru a contribui astfel la creșterea productivității și a calității produselor program rezultate;

• supravegherea consistentă presupune ca activitățile cuprinse în tehnologia de realizare a sistemului teleinformatic și referitoare la dezvoltarea de programe să fie:

— partiționate

— definit pentru fiecare activitate un criteriu de intrare, validarea și un criteriu de ieșire

— realizat un control statistic permanent;

• fiecare lucrare intermediară înainte de includerea ei în produs sau de-a i se asocia informația reală să se valideze;

• eventualele anomalii care apar la unele produse, pentru a putea ulterior fi analizate în vederea înlăturării atît a cauzelor care le-au generat cît și a efectelor pe care le pot produce, se înregistrează;

• schimbările, ivite pe parcurs la un produs program sau în activitatea de dezvoltare a acestuia, trebuie controlate;

• efectuarea de analize periodice care să stea la baza îmbunătățirii produselor program și a activităților legate de dezvoltarea acestora.

O primă versiune de arhitectură, pe care am putea-o numi *arhitectură curentă*, ar putea să se refere la activități bine definite în dezvoltarea de pro-

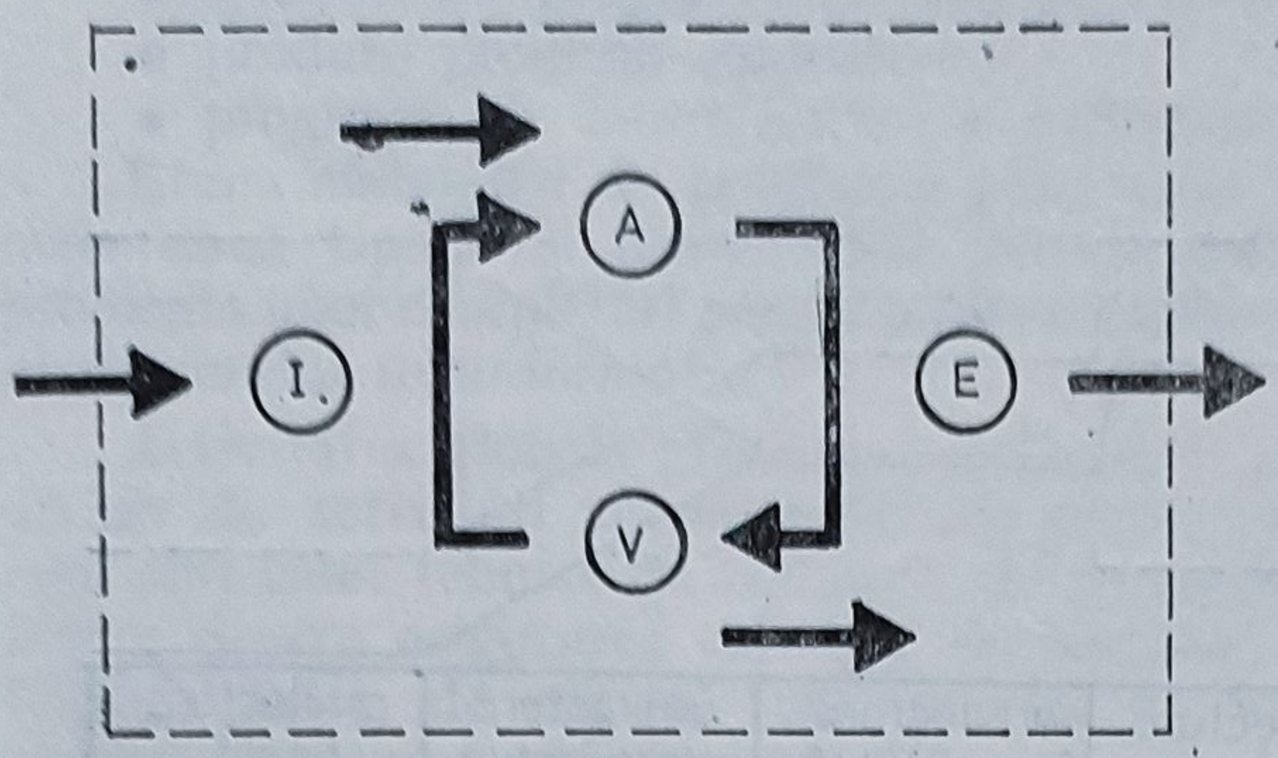


Fig. 4.17. Modelul IAVE

grame care în final să permită ca programele să fie utilizate pe o familie largă de calculatoare prin:

- transfer facil al produsului soft de pe un model de calculator pe altul
- asistarea în procesul de gestiune al produsului
- suport care să contribuie la dezvoltarea utilizării practice
- creșterea calității și productivității produselor software.

Arhitectura curentă reprezintă cel mai bun set de alternative (din cele „n" deja probate) pentru dezvoltarea software-ului în condițiile în care ele sînt disponibile pe sistemele de calcul care intră în realizarea rețelei de calculatoare.

Arhitectura trebuie să rămînă independentă de schimbările care pot surveni în metodologia, instrumentele și tehnicile folosite deoarece acestea referă modul cum trebuie făcut ceea ce trebuie făcut. Arhitectura nu conține detalii pentru activitățile de zi cu zi. Structura arhitecturii trebuie să permită urmărirea și controlarea dezvoltării produselor soft la calitatea lor cea mai ridicată.

Arhitectura curentă reprezintă primul pas în evoluția către o arhitectură finală. Atît arhitectura curentă cît și cele intermediare și cea finală au la bază un model, care reprezintă esența arhitecturii, și vizează activitățile din cadrul tehnologiei utilizată în dezvoltarea de programe. Modelul — pe care să-l numim IAVE — indică legăturile și fluxul între cele patru aspecte ale unei activități și este prezentat în fig. 4.17.

Pentru o activitate dată trebuie predefinite următoarele:

(1) o listă de criterii de intrare care ar trebui să fie îndeplinite înainte de începerea activităților de execuţie (I)

(2) un set de descrieri ale activităților de execuție care să indice ce trebuie făcut (A)

(3) o procedură de validare care să verifice calitatea lucrului produs de activitățile de execuţie (V)

(4) o listă verificată de criterii de ieșire care ar trebui să fie satisfăcută înainte ca activitățile de execuție să fie terminate (E).

Modelul IAVE nu implică derularea succesivă a activităților, multe dintre acestea putîndu-se efectua în paralel. Totuși pentru ca toate activitățile aparținînd unei etape să fi fost parcurse cu succes trebuie ca toate criteriile de ieșire să fi fost satisfăcute într-un anume punct în ciclul de viață al produsului. Dacă o procedură de validare indică o anomalie atunci bucla

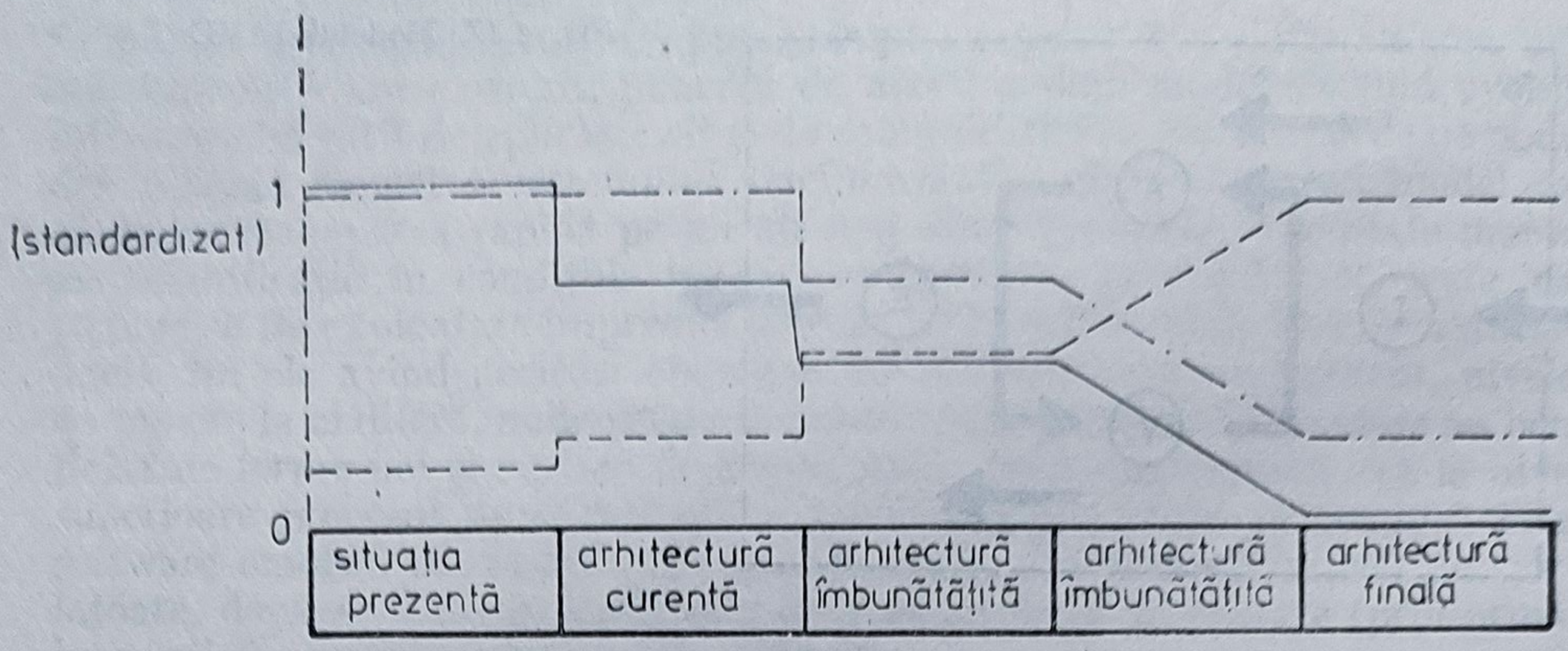


Fig. 4.18. Contribuția arhitecturii dezvoltării de programe la îmbunătățirea produselor program:

——— erori la mia de linii de cod;
– – – productivitatea măsurată în mii de linii de cod scrise de un programator pe lună;
—·—·— lungimea proiectului.

iterativă a activităților de execuție și validare este urmată pînă cînd se verifică că activitățile de execuție sînt finalizate în mod satisfăcător.

Modelul înfățișat în fig. 4.17 poate fi aplicat la diferite nivele de detaliu putînd contribui și la controlarea procesului de dezvoltare de programe. Produsul program aflat în dezvoltare poate fi monitorizat în fiecare moment al dezvoltării sale, prin aceasta obținîndu-se un înalt grad de control; se pot include acțiuni corective cerute de produs care să se realizeze înainte ca totul să se termine, și apoi să se determine nivelul de calitate probabil.

Pentru îmbunătățirea calității produselor program elaborate și destinate unui sistem teleinformatic se poate acționa în direcția micșorării erorilor raportate la o mie de linii de cod, crescînd productivitatea (exprimată în acest caz prin mii de linii de cod scrise de un programator pe lună) și prin micșorarea lungimii proiectelor.

O legătură a arhitecturii cu indicatorii evidențiați mai sus se prezintă în fig. 4.18.

Implementarea arhitecturii dezvoltării de programe conduce imediat la ridicarea calității produselor program.

4.5.2. ELABORARE DE PROGRAME

Etapa elaborării de programe își are obiectivele ei concrete pe care le urmărește: proiectarea și codificarea programelor, elaborarea și testarea programelor, elaborarea documentației de întreținere (pentru programe și date) precum și pregătirea testării.

Partea de software dintr-un sistem teleinformatic se compune din:

- sisteme de operare (care pot fi aceleași sau pot diferi pe sistemele de calcul aparținînd rețelei de calculatoare);
- sisteme de gestiune a teleprelucrării și sistem de gestiune a întregii rețele (software de comunicație).

- sisteme de programe pentru gestiunea datelor;
- produse program aplicative;
- programe de administrare a întregului sistem teleinformatic.

Etapa elaborării de programe pentru un STI vizează în principal ultimele două tipuri de componente, pentru restul punîndu-se eventual doar problema unor completări pentru a răspunde mai bine caracterului „pe măsură" al sistemului teleinformatic.

Activitatea globală privind elaborarea de programe cuprinde la rîndul ei un șir de activități corespunzătoare pașilor ce trebuie parcurși, activități cum sînt cele: tehnice de realizare, de pregătire a testării/testare, de control intern pentru asigurarea calității, de realizare a formei finale a produsului teleinformatic, de control (normativ și metodologic) efectuate la sfîrșitul elaborării tuturor programelor.

În etapa elaborării de programe a procesului tehnologic activitățile sînt stimulate de intrări diverse, în cele mai multe din cazuri ele fiind de tip specificații; derularea activităților conduce la o serie de produse intermediare (ca de exemplu: textul sursă, raportul de testare, fișiere, programe etc.) care reprezintă ieșirile. Tabelul 4.5 prezintă, pentru faza elaborării de programe relația intrări-activități-ieșiri.

Între activitățile tehnice de realizare se numără: studierea specificației de realizare a componentelor, proiectarea programelor, codificarea programelor/modulelor, codificarea componentelor care asigură integrarea programelor la nivel de proceduri de execuție, elaborarea documentației de întreținere.

Proiectarea fiecărui program începe cu identificarea și gruparea funcțiilor pe care trebuie să le îndeplinească programul, plasarea acestora în substructuri și stabilirea relațiilor de tip ierarhic existente între ele. Substructurile reprezintă o descompunere intermediară și nu includ nivelul de definire din cadrul proiectării la nivel de modul. Se denumesc modulele. Se specifică principalele structuri de date pentru fiecare modul precum și modul de control al ierarhiei dintre module.

Proiectarea la nivel de modul presupune o soluție logică specifică pentru fiecare modul. Se definesc structurile de date în cel mai mic detaliu. Modulele reutilizabile trebuie precizate.

Transpunerea algoritmilor de prelucrare, a structurii datelor, interfețelor utilizînd sintaxa și semantica limbajului de programare specificat, editarea modulelor de program pe suport magnetic, translatare (compilare, asamblare, interpretare etc.) actualizarea textului sursă sînt toate activități aparținînd codificării programelor/modulelor.

Activitățile de pregătire a testării presupun: proiectarea programelor auxiliare utilizate pentru testare și depanare, codificarea programelor auxiliare, stabilirea cazurilor de test pentru testul individual, stabilirea procedurii de testare pentru testul individual, efectuarea testării la nivelul testului individual, stabilirea cazurilor de test și a procedurii de testare pentru testul de integrare componentă funcțională.

Produsele finale ale etapei privind elaborarea de programe sînt documentația de întreținere, fișierele, bibliotecile de module/programe testate, specificația de testare pentru testul individual și listingurile martor cu rezultatele testului individual.

TABELUL 4.5. Activități (asociate cu intrări și ieșiri) în faza elaborării de programe

Intrări \ Ieșiri	Specificații de realizare program/modul	Text sursă	Program/modul	Text sursă procedură de execuție	Documentația de întreținere	Specificații de realizare pentru programe auxiliare	Specificația de testare pentru testul individual	Raport de testare	Date de test	Specificația de testare pentru testul de integrare componentă funcțională	Produse intermediare (validate intern)	Documentația de întreținere	Fișiere/Biblioteci de module/Programe testate	Specificația de testare pentru testul individual	Listinguri martor cu rezultatele testului individual	Raport de evaluare etapă
Specificații de realizare la nivel de componentă funcțională, SABD, proceduri de execuție	Activități tehnice de realizare															
Planuri de testare pentru testul individual și testul de integrare componentă funcțională						Activități de pregătire a testării/testare										
Specificații de testare, specificația de proiectare a testării																
Specificații de realizare program/modul																
Produse intermediare											Activități de control					
Produse intermediare (validate intern)												Realizarea formei finale a produsului				
Documentația de întreținere																Activități de control normativ și metodologic efectuate la sfîrșitul etapei
Fișiere/biblioteci de module/programe testate																
Specificația de testare pentru testul individual																
Listinguri martor cu rezultatele testului individual																

Documentaţia de întreţinere se elaborează la nivel de componentă funcţională şi cuprinde patru secţiuni. Prima secţiune prezintă modelul de ansamblu al sistemului teleinformatic (preluat din Proiectul de ansamblu actualizat cu modificările survenite pe parcursul realizării şi punerii în funcţiune a sistemului). De asemenea se prezintă modelul funcţional şi arhitectura configuraţiei de echipamente.

Complexitatea configuraţiei de echipamente, punctată şi de elemente cum sînt modemurile, liniile de comunicaţii, controloarele de reţea, terminalele care dau posibilitatea accesului de la distanţă a utilizatorului la întregul sistem teleinformatic, determină complexitatea întreţinerii. Prima secţiune se încheie cu lista produselor program de bază, a celor privind partea de comunicaţii de date precum şi a produselor program aplicative care determină în ansamblul lor funcţionarea sistemului.

Secţiunea a doua prezintă documentaţia de întreţinere pentru produsele program, care cuprinde specificaţia structurii produselor şi a documentaţiei finale şi descrierea produselor program (informaţii generale, destinaţia funcţională, descrierea structurii cu trimiteri la textele sursă, mijloacele tehnice utilizate, moduri de apelare, date de intrare/ieşire).

Secţiunea a treia vizează descrierea datelor, mai precis descrierea structurii datelor din baza de date sau din fişierele sistemului, la nivel de detaliu considerat necesar pentru întreţinerea aplicaţiilor. Fişiere distribuite în întregul sistem teleinformatic sau bazele de date distribuite cresc gradul de dificultate al întreţinerii aplicaţiilor.

Ultima secţiune se opreşte asupra instrucţiunilor de întreţinere a echipamentelor care intră în configuraţia sistemului teleinformatic. Se precizează instrucţiuni pentru întreţinerea individuală a echipamentelor precum şi instrucţiunile pentru exploatarea în ansamblu a tuturor echipamentelor. De exemplu aici se pot da indicaţii cu privire la modul de utilizare a testoarelor destinate comunicaţiilor de date care pot fi eficiente în menţinerea în condiţiuni de bună funcţionare a echipamentelor.

În etapa elaborării de programe, tehnologia de realizare a sistemului teleinformatic foloseşte metode, tehnici şi instrumente variate între care se numără: programarea structurată, programarea orientată pe fluxul datelor, programare software orientată pe obiecte, tehnici de învăţare, tehnica Wirth, tabele de decizie, programare funcţională, programare logică, metoda bazată pe planificarea rezolvării problemelor; lista prezentată mai sus rămîne deschisă.

Mergînd pe linia oferirii de exemple pentru fiecare etapă a tehnologiei de realizare a unui sistem teleinformatic din acelaşi domeniu al dispecerizării unei resurse naturale în paragraful următor se prezintă pachetul de programe SIMONE.

4.5.3. PACHETUL DE PROGRAME SIMONE

Pachetul de programe SIMONE (SI-mulation of M-emory O-ptimized NE-tworks and r-econstruction) [V-1] serveşte pentru controlarea/simularea proceselor dinamice care au loc într-o reţea de conducte destinată transportului şi distribuţiei gazului natural.

Rețeaua de transport și distribuție a gazului este formată din elemente diferite, interconectate, în număr de cîteva sute. Elementele de tip conductă sînt: tuburi, tuburi cu volum și rezistență zero (tăiate), tuburi cu volum mare și rezistență zero (recipient de gaz); elemente care nu sînt de tip conductă sînt: supape, stații reductoare, stații compresoare cu compresor piston, stații compresoare cu turbo-compresor, alte elemente (limitator de flux). Rețeaua este echipată și cu instrumente de măsură (de exemplu pentru presiune și viteza fluxului).

Prin „stare" vom înțelege atît descrierea valorilor de presiune și viteză a fluxului în toată rețeaua cît și informații despre modul real de lucru al tuturor instrumentelor de măsură și a echipamentelor tehnologice.

Pachetul de programe SIMONE poate fi folosit și pentru restabilirea stării întregii rețele.

Software-ul SIMONE poate servi la rezolvarea următoarelor probleme:

- obiectivizarea luării deciziilor în dezvoltarea și construcția unei rețele de conducte de distribuție a gazului metan;
- proiectarea unei structuri optimale a rețelei în concordanță cu cerințele pentru transportul gazului;
- dimensionarea optimală a conductelor și a echipamentelor tehnologice (în principal stațiile compresoare);
- dislocarea optimă a instrumentelor de măsură;
- instruirea personalului din centrul de control;
- pregătirea unei soluții pentru cazurile de avarie;
- oferirea de soluții pentru transportul gazului pe durata reparării unor echipamente tehnologice;
- controlul automat al transportului și distribuției gazului;
- optimizarea transportului gazului.

Lucrul cu pachetul de programe SIMONE poate fi caracterizat în termeni generali prin organigrama din fig. 4.19.

După definirea configurației rețelei de transport și distribuție a gazelor (care poate arăta ca cea din fig. 4.20) se trece la precizarea condițiilor pentru executarea simulării/controlului. Această etapă permite pentru cazul simulării schimbarea cererilor de furnizare de gaz, schimbarea modului de lucru a diferitelor echipamente tehnologice (închiderea/deschiderea supapelor, fisurarea unor conducte), prescrierea unor valori pentru presiuni în anumite puncte din rețea etc.; se pot fixa timpii de început și de sfîrșit pentru fiecare din acțiunile menționate mai sus. Tot pentru simulare se pot face precizări privind de exemplu stațiile compresoare, care se pot controla autonom — după prescrierile utilizatorului — în funcție de o presiune de ieșire constantă, raport de compresie constant, viteza fluxului gazului constantă. Pachetul reprezintă un instrument eficient pentru optimizarea transportului gazului natural.

Pentru cazul cînd se lucrează în modul control definirea condițiilor de control este implicită prin valorile culese folosind o rețea de calculatoare de tipul celei prezentată în paragraful 4.3.2 cînd s-a continuat studiul de caz prezentat în paragraful 4.2.3.

După terminarea simulării/controlului se pot evalua rezultatele și aceasta se face în regim de dialog, utilizînd un terminal grafic pentru partea de afișare. Se poate cere de exemplu afișarea presiunilor și a vitezei fluxului în

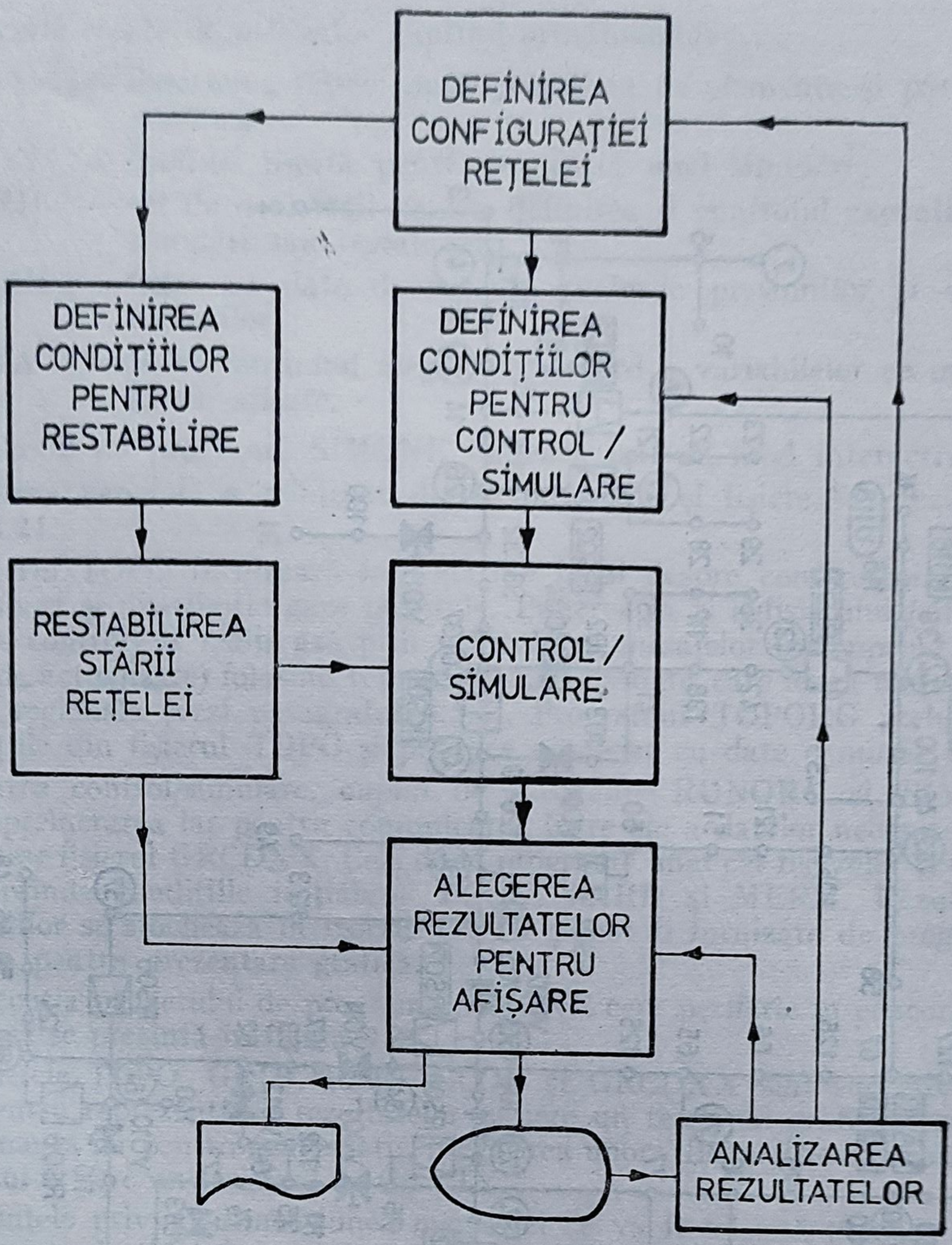


Fig. 4.19. Organigrama generală a pachetului de programe SIMONE

orice punct al rețelei; se poate alege scala pentru detaliere și eventual copierea ecranului la o imprimantă grafică. De asemenea se pot vizualiza caracteristici legate de funcționarea compresoarelor.

Bazat pe rezultatele obținute utilizatorul poate relua simularea/controlul în noile condiții. Prin simulare dinamică se pot găsi soluții la multe din problemele dificile ce apar în exploatarea curentă a rețelei de transport și distribuție gaze naturale.

Pachetul de programe SIMONE constă din patru programe principale: TOPORG, RUNORG, SIMRUN, RESDIS. Aceste programe cooperează cu nouă fișiere de date: TOPO, BCON, CARD, MERM, DIVA, GRIB, ICON, OUTP, GRCDXX, din care primele cinci trebuie create de utilizator, iar pe restul le creează pachetul de programe.

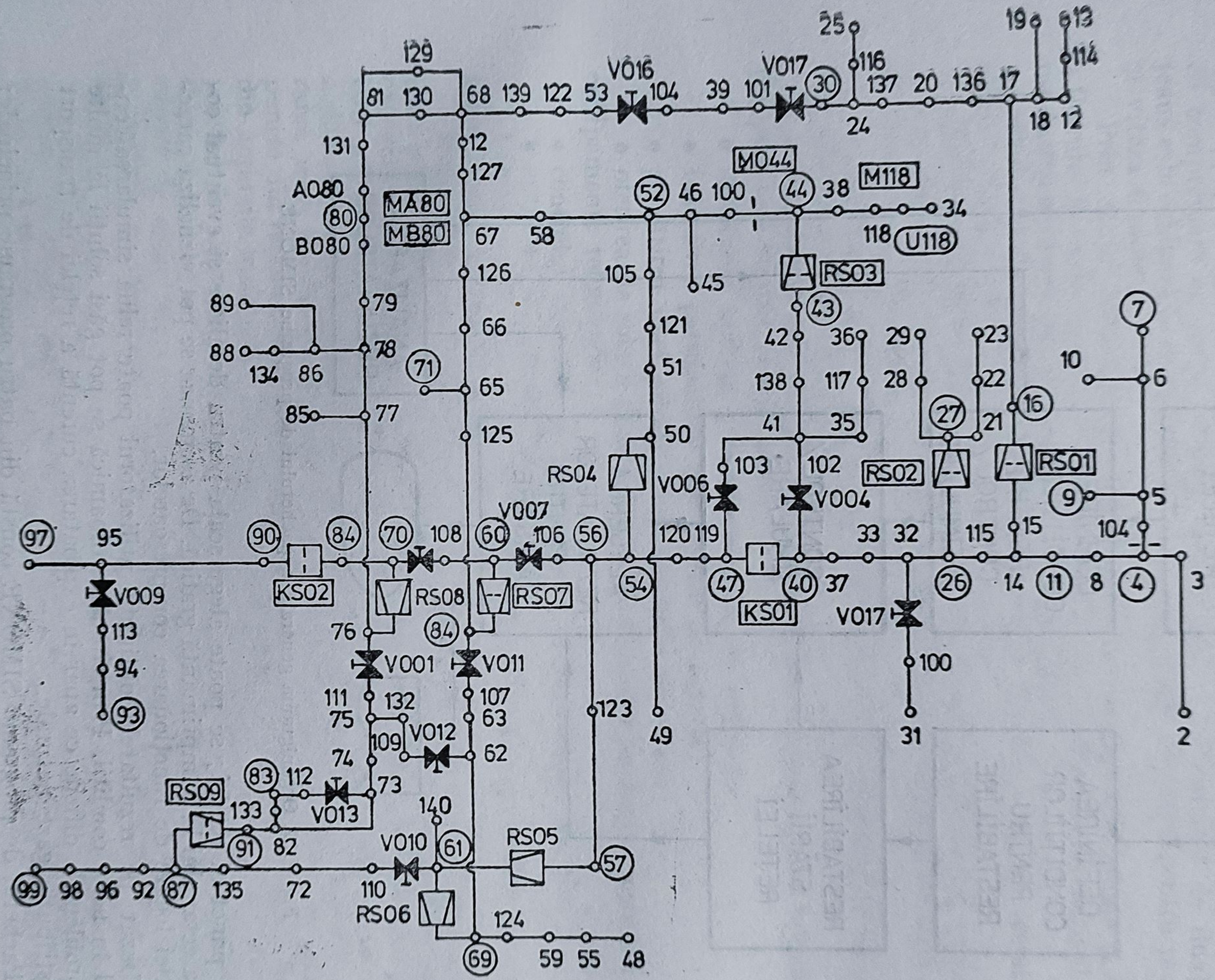


Fig. 4.20. Exemplu de configurație pentu o rețea de transport și distribuție a gazelor:
□ măsurarea fluxului; ○ măsurarea presiunii.

Fișierele create de utilizator cuprind următoarele:

TOPO — descrierea rețelei dată prin lista de elemente și parametrii acestora;

BCON — condiții limită pentru execuția unei simulări;

CARD — set de declarații pentru definirea și controlul execuției unei simulări sau restabiliri;

MERM — fișier cu date de măsură (valorile presiunilor și vitezele fluxurilor);

DIVA — fișier conținînd alegerea standard a variabilelor ce urmează să fie afișate.

Pachetul de programe SIMONE lucrează într-un mod interactiv.

Schema generală a relațiilor dintre programe și fișiere se prezintă în figura 4.21.

Fișierul TOPO furnizează informațiile la zi despre configurația rețelei de transport și distribuție gaze naturale. Punerea la zi a fișierului (situat în nodul de control) se realiz ază prin vehicularea mesajelor (ce cuprind informațiile de actualizare) folosind rețeaua de calculatoare care leagă toate dispeceratele regionale (vezi paragraful 4.3.2). Programul TOPORG prelucrează informațiile din fișierul TOPO și produce un fișier cu date denumit GRIB.

Pentru control/simulare, cuplul de programe RUNORG și SIMRUN asigură prelucrarea iar pentru comunicarea între ele a datelor necesare acestea folosesc fișierul GRCDXX. Cele două programe mai cer fișierele: ICON — care cuprinde condițiile inițiale —, BCON, CARD și MERM. Rezultatele prelucrărilor se stochează în fișierul OUTP și pot fi furnizate de programul RESDIS pentru prezentare grafică.

Execuția pachetului de programe SIMONE cere periferie în concordanță cu ceea ce se prezintă în figura 4.21.

Fișierele TOPO, GRIB, ICON, BCON și GRCDXX sînt succesiv activate. Pentru reprezentarea rezultatelor se cere un terminal grafic precum și o imprimantă iar pentru conversație, pregătirea unora dintre fișiere și controlul sistemului se cere un terminal de tip TTY.

Cerințele privind dimensiunea memoriei ce va fi folosită precum și cerințele legate de timpul de răspuns depind pe de-o parte de mărimea rețelei de distribuție și transport a gazului natural cît și de tipul de calculator folosit. Iată un exemplu: pentru o rețea de transport și distribuție a gazului natural cuprinzînd 146 noduri și 154 elemente (aici numărîndu-se 2 stații compresoare, 9 reductoare și 19 supape) s-a realizat o simulare pe un calculator EAI Pacer, folosindu-se 58 Kbytes memorie. Simularea funcționării acestei rețele timp de 244 ore a cerut doar 70 secunde.

Cît privesc cerințele software se menționează că subprogramele care depind de tehnica utilizată (de exemplu: fișierele de I/E, reprezentarea grafică etc.) trebuie adaptate de specialiști care stăpînesc tehnica de calcul de care se dispune și care pot exploata toate posibilitățile tehnice ale acestor echipamente.

Pachetul de programe SIMONE, considerat a fi unul dintre cele mai complexe realizat pînă în prezent, cunoaște multiple implementări.

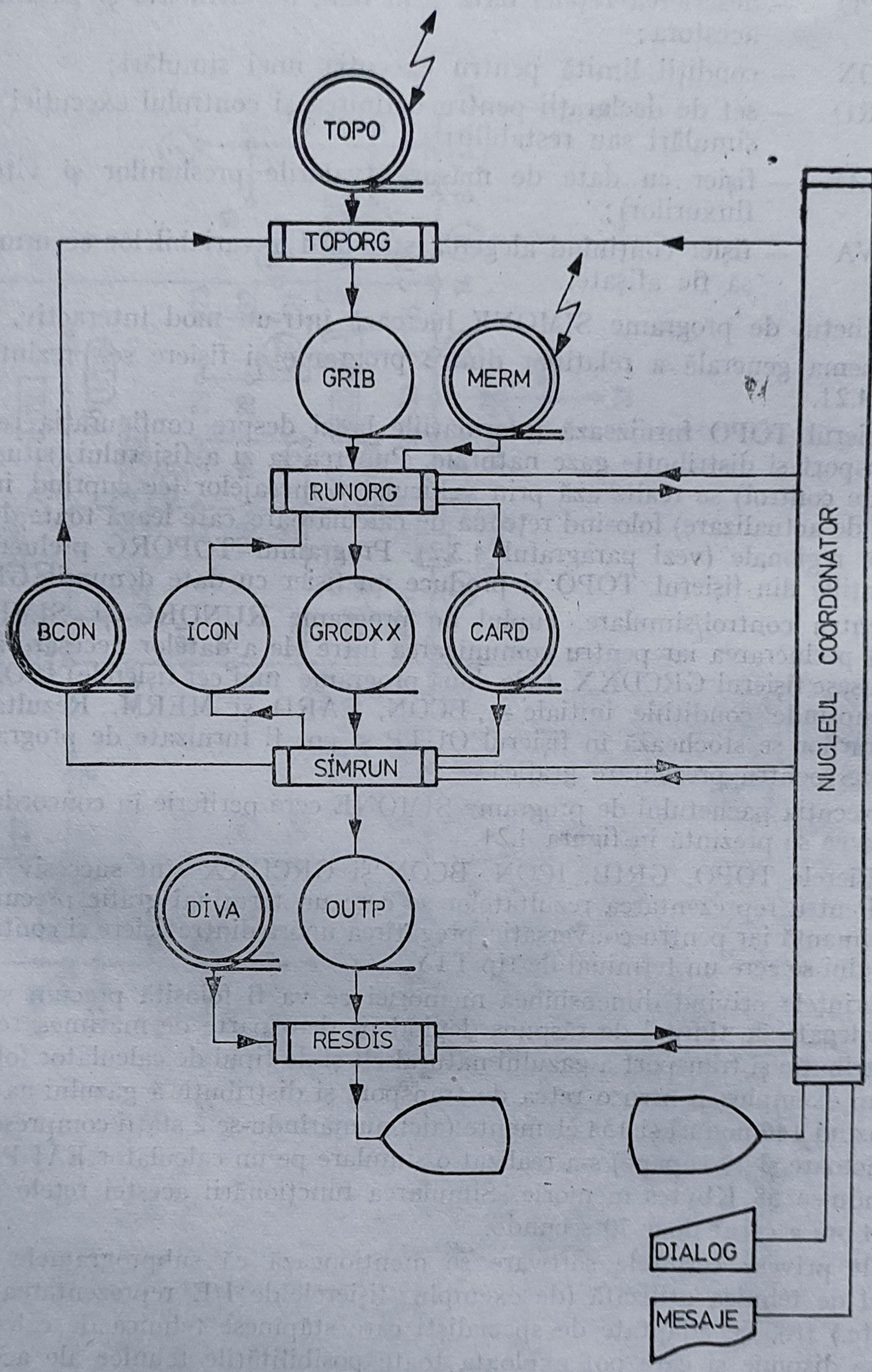


Fig. 4.21. Componente și legături funcționale în pachetul de programe SIMONE

4.5.4. INTEGRARE ȘI TESTARE

Etapa integrare și testare vizează atît componentele funcționale ale sistemului teleinformatic cît și întregul sistem.

Etapa își are obiectivele ei concrete pe care le urmărește: integrare și testare componentă funcțională, integrare și testare sistem, elaborarea documentației finale, planificarea punerii în funcțiune a sistemului/componentelor lui funcționale, pregătirea și efectuarea testării la nivel de integrare componentă funcțională/integrare sistem/la nivel de sistem precum și pregătirea recepției.

Activitatea globală de integrare și testare a unui sistem teleinformatic cuprinde o suită de activități (corespunzătoare pașilor ce trebuie parcurși și anume activități: tehnice de realizare, de planificare a procesului de realizare, de pregătire a testării/testare, de autocontrol (pentru asigurarea calității), de realizare a formei finale a produsului etapei, de control efectuate la sfîrșitul etapei. Stimularea activităților se obține cu ajutorul unor intrări diverse. Legăturile dintre intrări-activități-ieșiri, pentru această etapă, sînt ilustrate în tabelul 4.6.

Activitatea tehnică de realizare include integrarea componentelor funcționale utilizînd suportul hardware și software specificat în etapele anterioare. Aici se realizează elementele care asigură interfețele dintre componentele funcționale și cele cu mediul/utilizatorul și se realizează efectiv integrarea.

Activitatea de planificare a punerii în funcțiune a sistemului teleinformatic conduce la elaborarea unui plan.

Activitățile de pregătire a testării și de testare includ: stabilirea cazurilor de test și a procedurii de testare (pentru testul de integrare sistem, pentru testul de sistem și cel de acceptanță), efectuarea testării (pentru testul de integrare sistem și testul de sistem) și pregătirea recepției (acceptării).

Testările ce se pot realiza au ca obiectiv verificarea funcțională, verificarea produsului teleinformatic sau verificarea sistemului teleinformatic.

Terminarea în bune condițiuni a nivelului de testare funcțional (care presupune execuția tuturor funcțiilor unui produs integrat cu alte produse) semnifică că funcțiile produsului sînt capabile să lucreze în condiții de simulare și cu anumite constrîngeri.

Terminarea în bune condițiuni a nivelului de testare a produsului (care presupune execuția tuturor funcțiilor produsului lucrînd în condiții reale și fără constrîngeri) semnifică faptul că produsul devine comercial. Testul se execută din perspectiva utilizatorului dînd o măsură a performanțelor și posibilităților de utilizare ale produsului teleinformatic.

Testul pentru verificarea sistemului teleinformatic presupune execuția realizată din perspectiva utilizatorului, a tuturor funcțiunilor produselor într-un context hardware și software integrat. Cu această ocazie se evidențiază caracteristici de bază pentru sistemul teleinformatic cum sînt capacitatea sa de transport și memorare precum și disponibilitatea sa. Pentru fiecare sistem teleinformatic se proiectează și realizează teste pentru timpii de răspuns, debitul eficace, taxa de utilizare, capacitatea sistemului de-a satisface apelurile, ușurința de extensie, securitatea informațiilor, disponibilitatea sa [M—1].

Existența unei rețele de transport care permite utilizatorului să acceseze direct un sistem de calcul, generează principalele elemente care diferențiază

TABELUL 4.6. **Activități (asociate cu intrări și ieșiri) în faza de integrare și testare**

Intrări \ Ieșiri	Biblioteci de programe și componente integratoare (la nivel de sistem)	Plan de punere în funcțiune a sistemului teleinformatic	Specificație de testare (pentru testul de integrare sistem, de sistem și pentru testul de acceptanță)	Fișiere cu date de test	Raport de testare	Listinguri martor (cu rezultatele testului de integrare sistem)	Actualizări ale componentelor software	Programa și metodica de recepție	Produse intermediare ale etapei (validate intern)	Documentația de utilizare-exploatare (manuale)	Documentația de întreținere	Biblioteci cu componentele software (în format executabil)	Specificația de testare și listinguri martor	Fișiere cu date de test	Raport de evaluare etapă/fază
Tema de realizare Proiectul de ansamblu Biblioteci de programe și componente integratoare (pentru toate componentele funcționale, inclusiv cele referitoare la comunicații de date și baze de date)	Activități tehnice de realizare														
Produse intermediare		Activități de planificare a procesului de realizare													
Specificații de testare Biblioteci cu componente software			Activități de pregătire a testării/testare												
Produse intermediare									Activități de autocontrol						
Produse intermediare										Realizarea formei finale a produsului etapei					
Produsul final															Activități de control efectuate la sfîrșitul etapei

un sistem teleinformatic de unul informatic. Se impune testarea faptului că toate componentele reţelei de transport sînt în stare de funcţionare normală şi că valorile de toleranţă admisă nu sînt depăşite (de exemplu rata de erori pe liniile de transmisie are o valoare admisibilă).

Alte teste care se pot efectua pot avea ca obiectiv să permită asigurarea unei extensii a reţelei, realizabilă în condiţii de sistem operaţional, fără perturbarea bunei funcţionări.

Există teste cu acţiune la distanţă ca, de exemplu, în cazul telebuclării liniilor de comunicaţie sau în teleîncărcarea programelor de test.

În testarea componentelor de transmisie se pot folosi indicatoare luminoase, contoare, echipamente specializate de test conectabile în paralel pe echipamentul ce trebuie testat. Mijloacele informatice cuprinse sau implicate în testare pot furniza rapoarte sau alarme privind diferite componente ale reţelei. Mai mult atunci cînd anumite valori relative la componentele de comunicaţie depăşesc praguri prestabilite se pot lansa automat programe de test şi se poate comanda afişarea statisticilor rezultate.

Îndosarierea rezultatelor testării şi a specificaţiilor de testare pe nivele de test, păstrarea datelor de test pe nivele de test precum şi construirea bibliotecilor cu componentele software în format executabil reprezintă activităţi legate de realizarea formei finale a produsului etapei.

În această fază se redactează şi forma finală a documentaţiei de utilizare şi exploatare (aici încadrîndu-se manualul de utilizare şi manualul de exploatare — la nivel de componentă funcţională şi manualul de prezentare — la nivel de sistem) precum şi o documentaţie de întreţinere.

Manualul de prezentare conţine informaţii generale relative la întregul sistem teleinformatic. El cuprinde schema generală a fluxului informaţional, arhitectura sistemului teleinformatic, schema funcţională globală a componentelor sistemului, informaţii generale asupra intrărilor şi ieşirilor din sistem, resursele necesare pentru exploatarea sistemului (aici o menţiune specială fiind necesară pentru suportul fizic folosit în comunicaţiile la distanţă), eficienţa economică.

Manualul de utilizare (realizat pentru fiecare componentă funcţională) conţine informaţii despre procedurile de colectare, pregătire şi transmitere a datelor, procedurile de introducere/validare a datelor de intrare, procedurile de utilizare şi interpretare a ieşirilor.

Manualul de exploatare (la nivel de componentă funcţională) se adresează în principal personalului de exploatare din unitatea de informatică implicată în sistemul teleinformatic. El poate cuprinde graficul de exploatare a întregului sistem teleinformatic, a aplicaţiilor, procedurile de instalare a software-ului de comunicaţie şi a aplicaţiilor, de operare în timpul execuţiei, proceduri de control şi validare a execuţiei.

Pentru studiul de caz prezentat în paragrafele anterioare, se indică în figura 4.22 un extras din manualul de exploatare adresat inginerilor de sistem, şi anume o parte din controlul reţelei de transport [C—3].

Odată cu terminarea etapei de integrare şi testare se recomandă întocmirea unui raport de evaluare care descrie sumar activităţile realizate, confirmă trasabilitatea (adică respectarea cerinţelor din etapele anterioare), evaluează (din punct de vedere al completitudinii, conformităţii, neambiguităţii) produsul etapei. Eventualele anomalii constatate precum şi stadiul soluţionării lor sînt menţionate; nu sînt excluse şi unele recomandări.

CONTROLUL REȚELEI

Inginerul de sistem poate controla rețeaua cu ajutorul unui program utilitar numit NCP — Network Control Program. Exemplificările privind modul de lucru cu NCP se fac pentru modul MCR.

Invocarea și ieșirea din serviciul de control al rețelei

Invocarea serviciului de control al rețelei se face în maniera:

```
> NCP
```

iar pentru ieșire se folosește CTRL/Z.

Serviciul NCP poate fi utilizat doar după ce software-ul de comunicație a fost încărcat. Adoptăm un punct de vedere local.

Încărcarea și pornirea rețelei

Încărcarea software-ului de comunicații implică între altele aducerea Executivului de Comunicații în memorie și punerea nodului în starea „ON". Pentru aceasta folosim fișierul de comenzi NETINS.CMD,

```
> CHD 1,54
> @NETINS
```

Pornirea efectivă a software-ului de rețea se face cu comenzile:

```
> NCP SET SYS
> NCP SET EXE STA ON
```

Cum liniile sistem nu au fost setate pe ON la generarea software-ului de comunicație, trebuie realizat acest lucru acum cu comanda

```
>NCP SET LIN DUP-0 STA ON TOP
```

Oprirea rețelei și descărcarea

Înainte de-a scoate din memoria calculatorului software-ul corespunzător părții de rețea trebuie anulată funcționarea liniilor sistem. Comanda arată astfel:

```
> NCP CLE LIN DUP-0 ALL
```

apoi pentru îndepărtarea software-ului de comunicație se lansează fișierul de comenzi NETREM:

```
> CHD 1,54
> @NETREM
```

Structura fișierului NETREM:

NETREM.CMD

```
NCP SET EXE STA OFF
NCP CLE SYS
UNL HT:
```

Câteva comenzi utile

Cunoașterea stării liniilor sistem:

```
> NCP SHO STA KNO LIN
```

Cunoașterea stării nodurilor:

```
> NCP SHO STA KNO NOD
```

:

Fig. 4.22. Extras dintr-un manual de exploatare (pentru studiu de caz)

4.5.5. PACHETUL DE PROGRAME NSL

Pachetul NSL* (Network Snap Logger) este destinat depanării în context rețea a integrării aplicațiilor care folosesc drept suport pentru comunicațiile de date rețeaua de calculatoare.

Dezvoltat pentru minicalculatoarele românești și lucrînd sub un sistem de operare compatibil RSX-11, pachetul NSL poate fi utilizat cu succes în studiul de caz prezentat în paragrafele anterioare.

* realizat în ICSIT-TCI [C-2]

Pachetul NSL urmărește înregistrarea unor informații de control transmise din anumite puncte ale unui task aplicație, puncte selectate pentru a ajuta la eventualele depanări legate de funcționarea în comun a tuturor taskurilor aplicație.

Informațiile de control sînt grupate într-un bloc, numit SNAP. SNAP-ul se poate pune în orice punct al unui task aplicație din rețea. Se oferă în mod deosebit informații despre operațiile de intrare/ieșire din aplicațiile care utilizează rețeaua și chiar informații despre alte activități specifice acestor aplicații.

Pachetul NSL este compus din 3 programe (implementate ca task-uri) cu următoarele funcții:

- NSL (Network Snap Logger) — înregistrează SNAP-uri transmise de taskurile de aplicație din rețea
- NLX (Network Language eXchange) — realizează o interfață operator — pachet NSL
- LTF (List Trace File) — asigură listarea fișierului în care s-au înregistrat SNAP-urile.

SNAP-urile se transmit task-ului NSL prin lansarea unei directive executiv (SDAT$), utilizîndu-se pentru sincronizare un indicator de eveniment (40-global). SNAP-urile se înscriu într-un fișier, fie acesta NETTRC.NSL, paginat, împreună cu o serie de informații referitoare între altele la momentul apariției SNAP-ului, a tipului SNAP-ului etc.

Task-ul NSL recepționează de asemenea comenzi de la task-ul NLX care controlează modul de operare al task-ului NSL.

Un SNAP are o lungime de 26 octeți cu următoarea structură: 4 octeți pentru informații privind starea I/E, 16 octeți pentru date de I/E sau alte date, 4 octeți nefolosiți, 1 octet pentru identificatorul procesului (de obicei *lun* conexiune/linie), 1 octet pentru codul SNAP-ului.

Task-ul NSL se instalează cu comanda:

> INS NSL/TASK = NSL

și se lansează în execuție cu comanda:

> RUN NSL

Task-ul NLX transmite task-ului NSL două comenzi: prima de terminare NSL și implicit de încheiere fișier trace, a doua de încheiere fișier trece curent și de deschidere a unei noi versiuni a acestui fișier.

După instalarea cu comanda MCR:

> INS NLX/TASK = ... NLX

taskul NLX se lansează în execuție prin una din formele:

> NLX EX pentru prima comandă sau
> NLX CL pentru execuția comenzii a doua.

Task-ul LTF asigură decodificarea și listarea SNAP-urilor conținute în fișierul NETTRC.NSL. SNAP-ul se listează într-un rînd cu următoarea structură:

T. pp xDDDDDD ssssss ssssss ddd ddd... ddd unde T reprezintă numele task-ului sursă, pp identificatorul procesului, x specifică pentru blanc terminarea unei operații de I/E iar pentru punct lansarea operației de I/E, DDDDDD reprezintă denumirea SNAP-ului, ssssss ssssss starea I/E în octal pe cuvînt, ddd... ddd reprezintă date de I/E sau alte date în octal pe caracter (ultimele 3 caractere pot avea semnificație specială).

Task-ul LTF decodifică codurile de SNAP-uri standard în denumiri standard. În tabelul 4.7 se prezintă spre exemplificare trei din cele treizeci de coduri prevăzute.

TABELUL 4.7. Exemple pentru coduri de SNAP

Denumire standard	Cod	Altă condiție	Semnificație
CONECT	001		cerere de conectare soft rețea
ACC	002	pp ne 0	acceptare conectare
REJ	002	pp eq 0	rejectare conectare

Task-ul LTF se instalează prin comanda MCR:

>INS LTF/TASK = ... LTF

Formatul general al comenzii de listare este:

>LTF [vv] [/hh: mm]

cu semnificația de listare a fișierului NETTRC. NSL; vv unde vv reprezintă ultima versiune a fișierului iar hh: mm semnifică faptul că trebuie listate mesajele începînd de la ora precizată. Fișierul NETTRC. NSL lucrează prin reacoperire.

4.6. PUNERE ÎN FUNCȚIUNE/EXPERIMENTARE ȘI ACCEPTARE SISTEM TELEINFORMATIC

Această etapă vizează atît componentele funcționale ale sistemului teleinformatic cît și întregul sistem. Ea are următoarele obiective concrete pe care le urmărește:

- acțiuni pregătitoare în vederea punerii în funcțiune (instruire personal, măsuri organizatorice și tehnice);
- punere în funcțiune/experimentare la unitatea beneficiară;
- efectuarea testării la nivelul testului de acceptanță/recepție;
- recepția sistemului teleinformatic;
- actualizarea documentației finale a sistemului/componentelor funcționale/sistemului de administrare a bazei de date/sistemului de gestiune a comunicațiilor.

Activitatea globală de punere în funcțiune/experimentare și acceptare sistem teleinformatic cuprinde activități tehnice, activități de testare/recepție, activități legate de realizarea formei finale a produsului etapei precum și activități de control efectuate la sfîrșitul etapei (recepția sistemului teleinformatic și întocmirea procesului verbal de recepție). Legăturile dintre intrări—activități—ieșiri pentru această etapă sînt ilustrate în tabelul 4.8.

Activitățile tehnice includ acțiuni pregătitoare în vederea punerii în funcțiune, punerea în funcțiune/experimentare la unitatea beneficiară și asistență tehnică la beneficiar. În instruirea personalului o atenție deosebită trebuie să se acorde personalului care va asigura exploatarea rețelei de comunicație iar măsurile organizatorice și tehnice trebuie să aibă în vedere dispersarea geografică a calculatoarelor participante la sistemul teleinformatic. Punerea în funcțiune la beneficiar a sistemului teleinformatic pre-

TABELUL 4.8 Activităţi (asociate cu intrări şi ieşiri) la punerea în funcţiune/experimentare şi acceptare sistem teleinformatic

Intrări \ Ieşiri	Actualizări la manuale	Fişiere cu date de test	Raport de testare	Listinguri martor	Actualizări ale componentelor software	Actualizări în Documentaţia de utilizare-exploatare şi Documentaţia de întreţinere	Biblioteci cu componentele software	Proces verbal de recepţie
Tema de realizare	Activităţi tehnice							
Plan de punere în funcţiune								
Manualul de utilizare								
Manualul de exploatare								
Documentaţia de întreţinere								
Biblioteci cu componentele software								
Fişiere cu date de test								
Plan de testare		Activităţi de testare/ recepţie						
Specificaţia de testare								
Documentaţia de utilizare-exploatare								
Documentaţia de întreţinere								
Biblioteci cu componentele software								
Documentaţia de utilizare-exploatare						Realizarea formei finale a produsului etapei		
Documentaţia de întreţinere								
Biblioteci cu componentele software								
Produsul etapei								Activităţi de control efectuate la sfîrşitul etapei Recepţia

supune: verificarea existenţei în toate unităţile beneficiarului participante la sistemul teleinformatic a suportului hardware şi software specificat, verificarea modului de obţinere a datelor de intrare din mediul utilizator, încărcarea şi verificarea software-ului de comunicaţie, restaurarea bibliotecilor/fişierelor cu componentele software de aplicaţie realizate, încărcarea fişierelor/bazei de date, lansarea în funcţiune a procedurilor de execuţie conform graficului de ordonanţare a execuţiei.

Activităţile de testare/recepţie presupun efectuarea testării la nivelul testului de acceptanţă/recepţie cu definitivarea procedurii de testare, generarea datelor de test, execuţia testelor şi depanare (dacă aceasta se impune).

Introducerea actualizărilor făcute în componentele software corespunzătoare precum şi efectuarea de actualizări în documentaţie contribuie la realizarea formei finale a produsului etapei.

Revenind la exemplul luat în discuţie pe parcursul capitolului, privind un sistem teleinformatic care să asiste activitatea de dispecerizare la nivel naţional a unei resurse naturale, se menţionează că pentru acceptarea sistemului teleinformatic trebuie ca în cadrul unor experimentări să se probeze cele trei mari funcţii ale exploatării: supravegherea bunei funcţionări a sistemului, asistarea în utilizarea sistemului şi intervenirea în caz de incidente.

Se verifică că toate componentele sistemului sînt în stare de funcţionare normală sau că cele cîteva toleranţe admise nu sînt depăşite (de exemplu că rata de erori pe liniile de transmisie este suportabilă). Prin testele care se pot lansa (de exemplu pentru asigurarea în privinţa calităţii unei componente puţin utilizată sau pentru întreţinerea preventivă a echipamentelor) se poate obţine în orice moment o imagine completă a stării întregului sistem

Utilizatorul sistemului trebuie pe de-o parte să poată fi sfătuit în utilizarea sistemului iar pe de altă parte să fie asistat în detectarea, localizarea şi ridicarea defectelor de orice tip.

Acceptarea sistemului teleinformatic trebuie să se facă după analiza performanţelor acestuia. O evaluare a performanţelor se face înainte ca sistemul să fie realizat, aceasta constituind o etapă importantă în faza de concepţie a lui. Cum performanţele caracteristice pentru un STI sînt numeroase, înaintea intrării în exploatare a acestuia se impune măsurarea acelor performanţe care par să fie mai importante; măsurătorile pot să se facă într-un punct, în ansamblul sistemului, să se efectueze direct sau să necesite o prelucrare autonomă. Măsurarea performanţelor unui STI în funcţiune devine una din sarcinile principale ale exploatării.

Exploatarea şi întreţinerea unui sistem teleinformatic reprezintă etapa din ciclul de viaţă al unui STI imediat următoare ultimei etape din ciclul celor şase care contribuie la realizarea sistemului. Funcţionarea curentă a sistemului la parametrii proiectaţi, întreţinerea sistemului şi actualizarea documentaţiei reprezintă principalele obiective ale etapei de exploatare şi întreţinere a unui sistem teleinformatic.

Calitatea tuturor activităţilor, metodelor, tehnicilor şi instrumentelor folosite în realizarea sistemului teleinformatic contribuie direct la asigurarea unei exploatări eficiente şi îndelungate a sistemului teleinformatic proiectat.

Cunoştinţe privind hardware-ul şi software-ul de calculatoare, de transmisii de date, mijloacele de transport a datelor şi tarifarea lor, exploatarea marilor sisteme, teoria cozilor de aşteptare, statistici, metode numerice avansate, simularea, optimalizarea reţelelor, calculul fiabilităţii, ergonomie, economie reprezintă elemente care trebuie stăpînite şi mai ales bine integrate în momentul realizării unui STI.

CAPITOLUL 5

PROIECTAREA SISTEMELOR DE PROTECȚIE ȘI AUTENTIFICARE ÎN STI

Cerințele de comunicații între calculatoare și echipamente ale diferitelor întreprinderi și instituții cresc continuu. Proiectarea unor sisteme teleinformatice trebuie să satisfacă cîteva condiții fundamentale de fiabilitate, protecție și securitate a informațiilor memorate în diverse puncte ale rețelei sau transmise pe liniile de comunicații.

Apariția și dezvoltarea unor rețele de calculatoare mari, în cadrul cărora calculatoarele electronice sînt instalate la distanță mare între ele, amplifică importanța protecției informațiilor și autentificarea utilizatorilor. Aceste probleme devin deosebit de complexe, atît sub raportul posibilităților de acces fizic la liniile de telecomunicație, cît și al acelora de intervenție în software-ul de rețea, întrucît pot facilita accesul la transmiterea sau modificarea unor date de către utilizatorii neautorizați.

O tratare exhaustivă a aspectelor securității și protecției datelor la sistemele de calcul în general se face în lucrarea [A-1].

5.1. OBIECTIVE ALE PROTECȚIEI INFORMAȚIILOR ÎN STI

Obiectivul principal al măsurilor de protecție într-un sistem teleinformatic îl constituie eliminarea posibilităților de distrugere accidentală sau voită a informațiilor, precum și de consultare neautorizată a acestora. Accesul neautorizat la informații poate provoca serioase daune prin afectarea caracterului privat al transmisiilor, introducerea unor date false sau trunchiate, falsificarea identității unor calculatoare, terminale, ori utilizatori. Dintre obiectivele importante ce trebuie avute în vedere la proiectarea unor mecanisme — hardware și software — pentru protecția informațiilor într-o rețea le menționăm pe cele de prevenire a dezvăluirii conținutului în clar al mesajelor, modificării selective a mesajelor, inserării de mesaje false, analizei traficului de mesaje, precum și pe cele de detectare a modificării, ștergerii sau înlocuirii conținutului mesajelor, încercărilor de conectare neautorizată în rețea.

Obiectivele de prevenire pot fi atinse prin *criptarea informațiilor*, soluție eficientă cînd memorarea sau transmiterea de date se efectuează pe medii nesigure. Obiectivele de detectare sînt realizate dacă se folosesc *protocoale*

specifice, coroborate cu metode criptografice, care asigură schimburile de mesaje între entitățile rețelei.

Criptarea conferă protecție informației transmise pentru canalele ce sînt ascultate sau interceptate. În acest scop emițătorul alege un algoritm de cifrare și o cheie, pe care le comunică receptorului pe cale sigură, — de exemplu, prin curier. Criptografia modernă protejează datele transmise pe linii de telecomandă de mare viteză și memorate în calculatoare. Ea urmărește două obiective principale, și anume: *protecția* (prevenirea dezvăluirii neautorizate a unor informații transmise sau memorate) și *autenticitatea* ori integritatea (prevenirea unor modificări neautorizate ale datelor).

5.2. METODE CRIPTOGRAFICE ȘI APLICAȚIILE LOR ÎN STI

Un *cifru* reprezintă transformarea unui *mesaj clar* sau text clar în *mesaj-cifrat* ori *criptogramă*. Procesul de transformare a textului clar în text cifrat se numește *cifrare* sau *criptare*, iar transformarea inversă, a criptogramei în text clar , are denumirea de *descifrare* ori *decriptare*. Atît cifrarea, cît și descifrarea sînt controlate de o *cheie* (fig. 5.1).

Cripto-analiza studiază metodele de spargere a cifrurilor, adică de determinare a textului clar sau a cheii de cifrare din criptogramă.

5.2.1. SISTEME CRIPTOGRAFICE

Un sistem criptografic (un criptosistem) are cinci componente:

— spațiul mesajelor în text clar, $\mathcal{M}$;
— spațiul mesajelor în text cifrat, $\mathcal{C}$;
— spațiul cheilor, $\mathcal{K}$;
— familia transformărilor de cifrare, E_K: $\mathcal{M} \rightarrow \mathcal{C}$; unde $K \in \mathcal{K}$;
— familia transformărilor de descifrare, D_K : $\mathcal{C} \rightarrow \mathcal{M}$, unde $K \in \mathcal{K}$.

Fiecare transformare de cifrare, E_K, este definită de un algoritm de cifrare, E, comun tuturor transformărilor familiei, și o cheie, K, distinctă de la o transformare la alta. În mod similar, fiecare transformare de descifrare, D_K, este definită de un algoritm de descifrare D, și de cheia K. Pentru un K dat, D_K reprezintă inversa lui E_K, adică:

$$D_K(E_K(M)) = M, \quad \forall M \in \mathcal{M}.$$

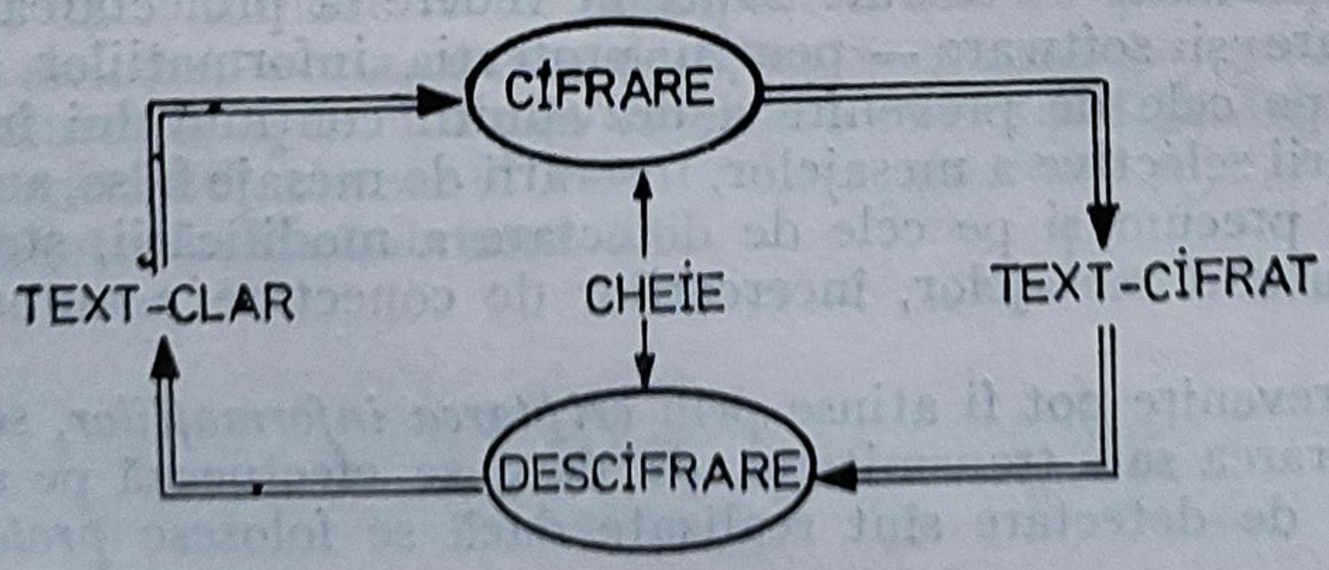


Fig. 5.1. Sistem de cifrare/descifrare

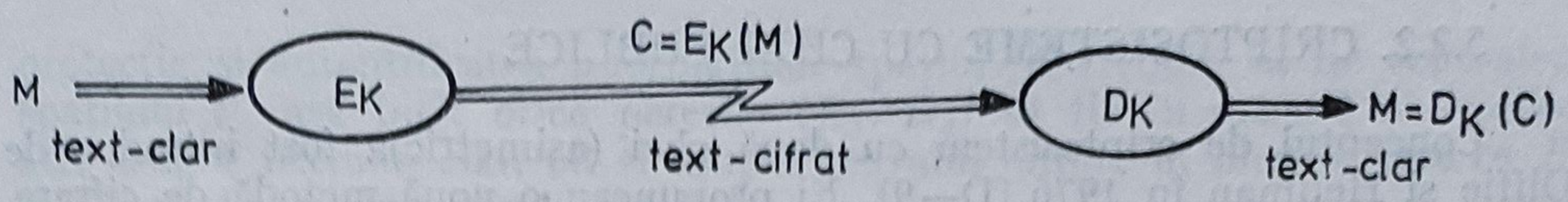


Fig. 5.2. Sistem criptografic

În figura 5.2. este ilustrat modul în care transformările de cifrare și descifrare sînt folosite pentru a se asigura protecția unui transfer de informații într-un STI.

Datele dintr-un STI trebuie astfel protejate încît utilizatorii neautorizați să nu poată reconstitui textul clar dintr-un text cifrat interceptat. În acest sens, trebuie să asigure ca:

— utilizatorul neautorizat să nu poată determina sistematic transformarea de descifrare, D_K, din textul cifrat, C, interceptat, chiar dacă se cunoaște textul clar, M, corespondent;

— utilizatorul neautorizat să nu poată reconstitui textul clar, M, din textul cifrat, C, fără cunoașterea transformării D_K.

Protecția datelor impune ca transformarea de cifrare D_K (respectiv, cheia) să fie protejată (fig. 5.3).

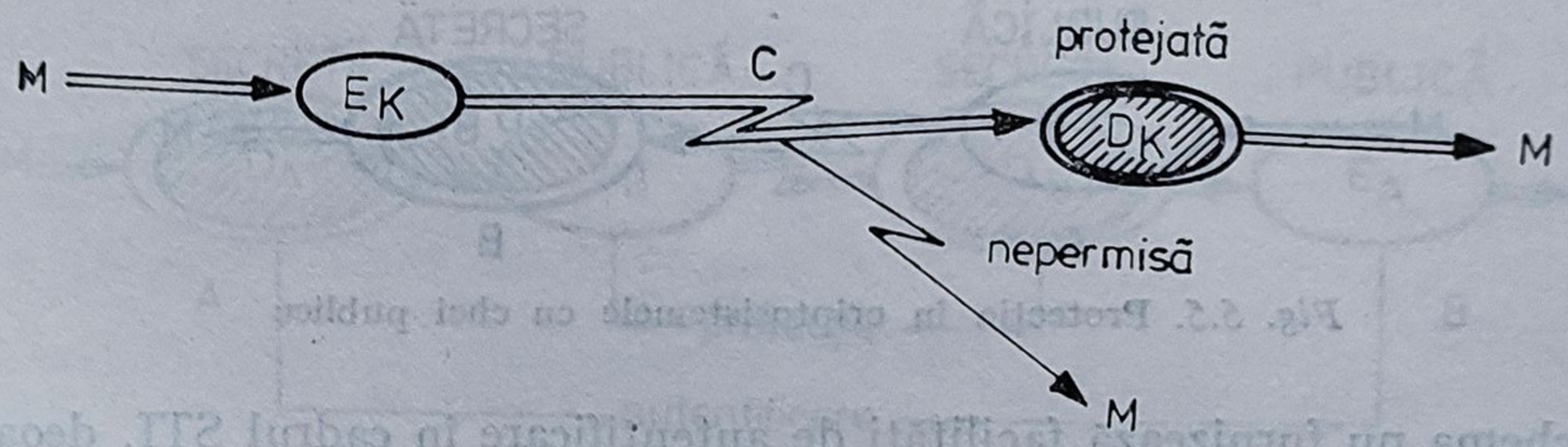


Fig. 5.3. Protecția datelor

Autentificarea datelor în STI cere ca un utilizator neautorizat să nu fie capabil în mod obiectiv să substituie textul cifrat, C, cu un text cifrat, fals, C', fără ca acest lucru să fie detectat. Nu trebuie să i se permită utilizatorului neautorizat:

— să determine sistematic transformarea E_K, cunoscînd pe C și textul clar corespunzător, M;

— să găsească în mod sistematic C', astfel ca $D_K(C')$ să fie un text clar valid în M. Cerințele de autentificare impun doar ca transformarea E_K (respectiv, cheia de cifrare) să fie protejată (fig. 5.4).

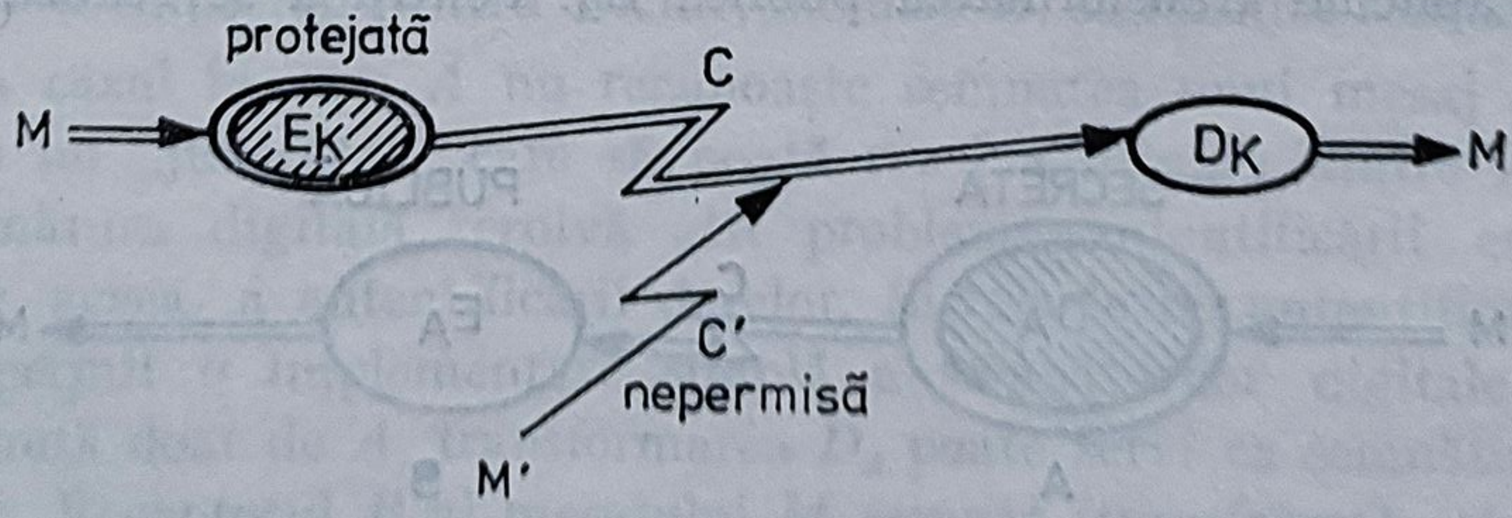


Fig. 5.4. Autentificarea datelor

5.2.2. CRIPTOSISTEME CU CHEI PUBLICE

Conceptul de criptosistem cu două chei (asimetric)a fost introdus de Diffie şi Hellman în 1976 [D—9]. Ei propuneau o nouă metodă de cifrare, numită cifrare cu cheie publică, în cadrul căreia doi utilizatori (procese) pot comunica cunoscînd fiecare doar cheia publică a celuilalt.

În criptosistemele cu chei publice fiecare utilizator A, al unui STI deţine o *transformare de cifrare publică*, E_A, care poate fi memorată într-un registru (fişier) public şi o *transformare de descifrare secretă*, D_A, ce nu este posibil să fie obţinută din E_A. Cheia de descifrare (secretă) este derivată din cheia de cifrare (publică) printr-o transformare *one-way*.

În criptosistemele cu chei publice, protecţia şi autentificarea sînt realizate prin transformări distincte. Să presupunem că utilizatorul (proces) A al unui STI doreşte să emită un mesaj, M, unui alt utilizator (proces) B. Dacă A cunoaşte transformarea publică E_B, atunci A poate transmite M la B sub forma $C = E_B(M)$, asigurîndu-se astfel funcţia de protecţie.

La recepţie, B, va descifra criptograma C utilizînd transformarea D_B, cunoscută doar de el:

$$D_B(C) = D_B(E_B(M)) = M.$$

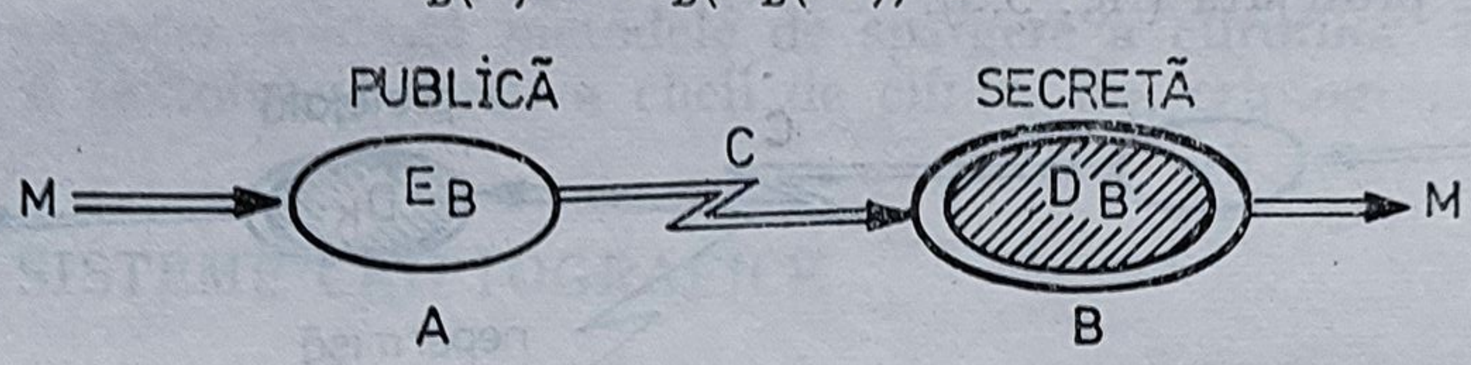


Fig. 5.5. Protecţia în criptosistemele cu chei publice

Schema nu furnizează facilităţi de autentificare în cadrul STI, deoarece orice utilizator (proces) are acces la transformarea publică E_B a lui B şi îi poate trimite mesaje false M' sub forma $C' = E_B(M')$.

Pentru autentificare se aplică lui M transformarea secretă D_A a lui A. Ignorînd protecţia pentru moment, A va emite $C = D_A(M)$ la B, care la recepţie va aplica transformarea publică, E_A a lui A:

$$E_A(C) = E_A(D_A(M)) = M.$$

Autentificarea este realizată deoarece numai A poate aplica transformarea D_A.

Protecţia nu este asigurată, întrucît este posibil ca M să fie obţinut de oricine aplicînd transformarea publică E_A. Pentru a se realiza simultan

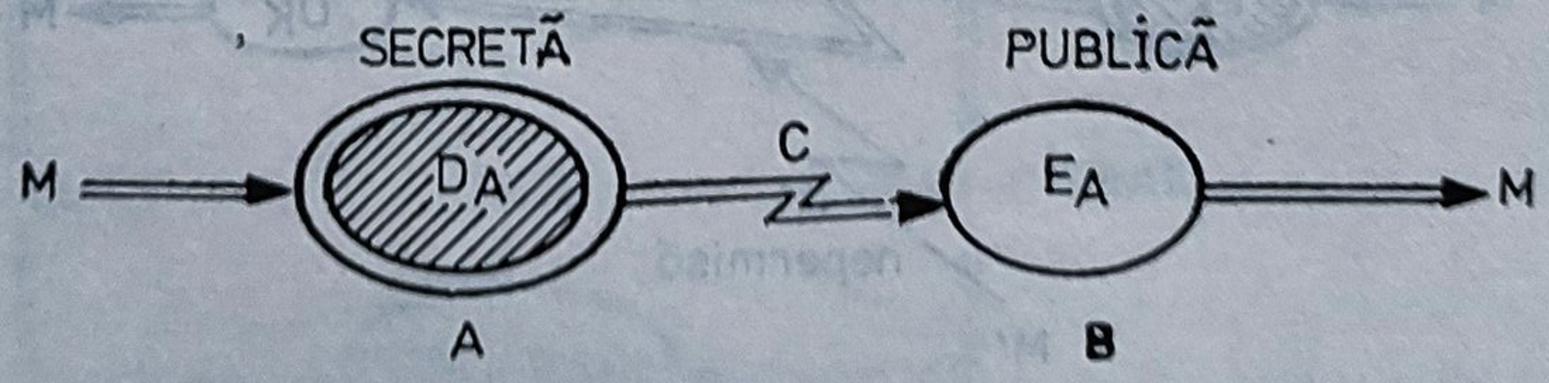


Fig. 5.6. Autentificarea în criptosistemele cu chei publice

protecția și autentificarea informațiilor spațiul $\mathcal{M}$ trebuie să fie echivalent spațiului $\mathcal{C}$, așa încît orice pereche (E_A, D_A) să fie în măsură să opereze atît asupra textului clar, cît și asupra textului cifrat; în plus se cere ca E_A și D_A să fie mutual inverse, adică:

$$E_A(D_A(M)) = D_A(E_A(M)) = M.$$

Emițătorul de mesaj A va aplica mai întîi transformarea secretă a sa, D_A, mesajului M. Apoi A va cifra rezultatul — utilizînd transformarea publică a lui B, E_B și va emite către receptor criptograma:

$$C = E_B(D_A(M)).$$

Receptorul B îl obține pe M aplicînd la început propria-i funcție de descifrare, D_B, iar apoi transformarea publică a lui A, E_A, cea care furnizează autentificarea:

$$\begin{aligned} E_A(D_B(C)) &= E_A(D_B(E_B(D_A(M)))) \\ &= E_A(D_A(M)) \\ &= M. \end{aligned}$$

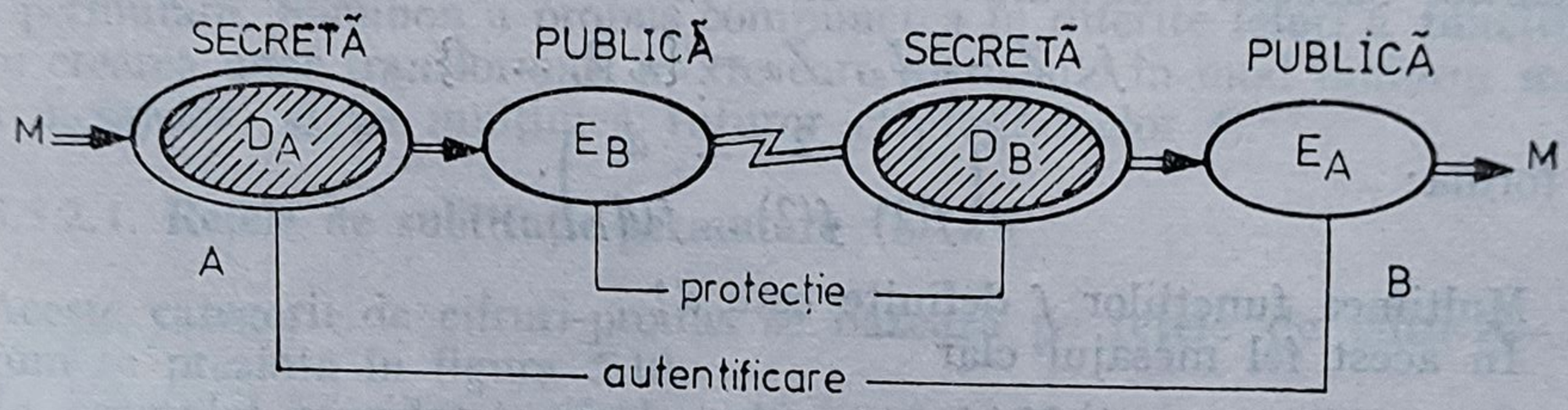


Fig. 5.7. Protecția și autentificarea în criptosistemele cu chei publice

5.2.3. SEMNĂTURA DIGITALĂ

Semnătura digitală reprezintă o proprietate a unui utilizator sau proces al unui STI și este utilizată pentru recunoașterea acestuia. Fie B un receptor de mesaj semnat de A. Semnătura lui A trebuie să satisfacă cîteva proprietăți:

— B să fie capabil să valideze semnătura lui A;

— să fie imposibil pentru oricine, inclusiv B, să falsifice semnătura lui A;

— în cazul în care A nu recunoaște semnarea unui mesaj M, trebuie să existe un „judecător" care să poată rezolva disputa dintre A și B.

Semnătura digitală rezolvă atît problema autentificării emițătorului cît și pe aceea a autentificării datelor. Sistemele de autentificare cu chei publice permit o implementare simplă a semnăturilor digitale. Deoarece este deținută doar de A, transformarea D_A poate servi ca semnătură digitală pentru A. Receptorul B al mesajului M semnat (transformat prin D_A) este sigur atît de autenticitatea emițătorului, cît și de aceea a datelor. Deoarece

transformarea inversă E_A este publică, receptorul B va putea valida semnătura. Procesele se desfășoară astfel:

— A semnează pe M, calculînd $S = D_A(M)$;

— B validează semnătura lui A, verificînd dacă $E_A(S) = M$;

— un „judecător" rezolvă eventuala dispută dintre A și B controlînd dacă $E_A(S)$ conduce la M, în aceeași manieră ca și B.

Implementarea semnăturii digitale este posibilă și prin utilizarea sistemelor de cifrare convenționale (de tip DES).

5.3. PROIECTAREA ALGORITMULUI DE CIFRARE COMPUTAȚIONALĂ

5.3.1. ALGORITMI DE BAZĂ

În această clasă se includ cifrurile transpoziție și substituție.

Cifrurile transpoziției realizează o permutare a caracterelor din textul-clar. Cheia de cifrare este perechea $K = (d, t)$, unde d reprezintă lungimea blocurilor succesive de caractere care vor fi cifrate conform permutării f,

$$f: Z_d \to Z_d, \quad Z_d = \{1, 2, \ldots, d\},$$

de forma: $\begin{pmatrix} 1 & 2 & \ldots & d \\ f(2) & f(2) & \ldots & f(d) \end{pmatrix}$.

Mulțimea funcțiilor f definite este $d!$

În acest fel mesajul clar

$$M = m_1 \ldots m_d m_{d+1} \ldots m_{d+2} \ldots$$

este cifrat astfel:

$$E_K(M) = m_{f(1)} \ldots m_{f(d)}, \quad m_{d+f(1)} \ldots m_{d+f(d)} \ldots$$

Descifrarea se obține prin permutarea inversă.

În cazul cifrurilor computaționale transpoziția se realizează prin așa-numitele cutii P (fig. 5.8).

Cifrurile transpozitive reprezintă componente importante în criptosistemele complexe care vor fi prezentate mai departe.

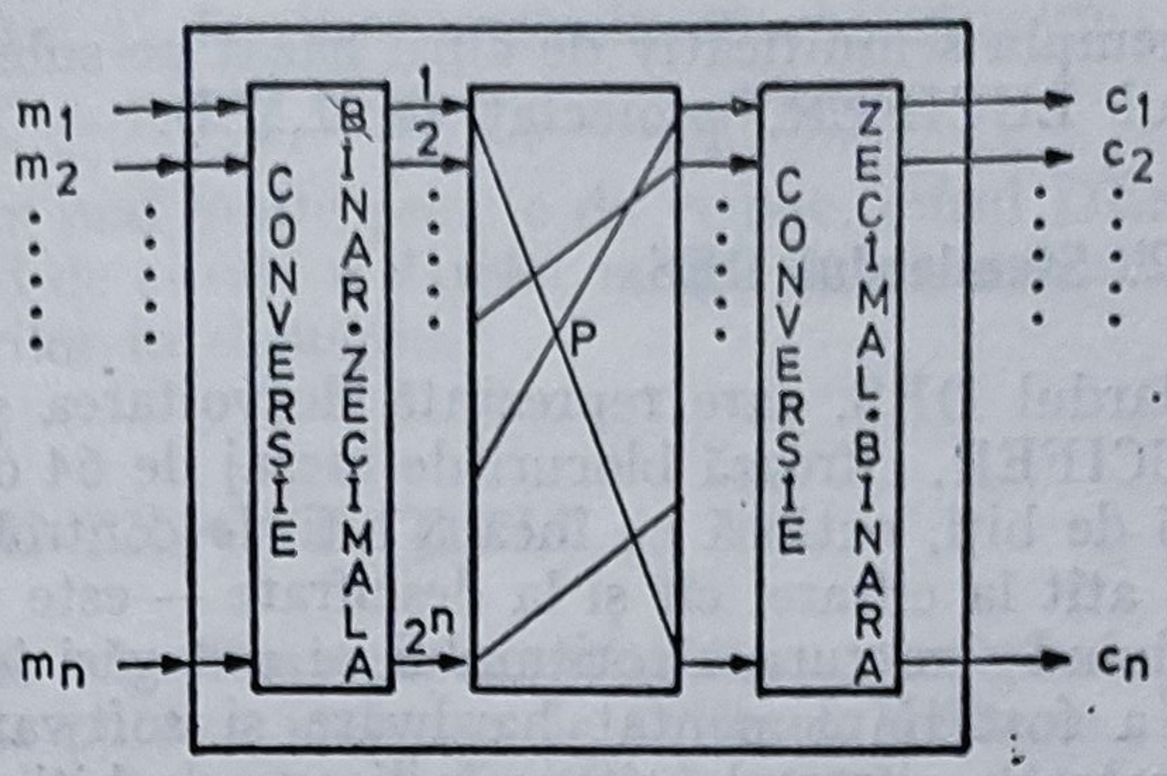
Cifrurile substituție înlocuiesc fiecare caracter din alfabetul mesajelor $\mathcal{A}$ cu un caracter corespunzător din alfabetul criptogramelor $\mathcal{C}$. Dacă $\mathcal{A} = \{a_1, a_2, \ldots, a_n\}$, atunci $\mathcal{C} = \{f(a_1), f(a_2), \ldots f(a_n)\}$, unde $f: \mathcal{A} \to \mathcal{C}$ este funcția care face să corespundă fiecărui caracter din $\mathcal{A}$, unul din $\mathcal{C}$. Cheia cifrului este dată de funcția f.

Cifrarea unui mesaj $M = m_1, m_2, \ldots m_n$ se face astfel:

$$E_K(M) = f(m_1) f(m_2) \ldots f(m_n).$$

Fig. 5.8. Cutie P pentru realizarea transpoziției

Fig. 5.9. Cutie *S* pentru realizarea substituției



În cadrul cifrurilor computaționale substituția simplă se realizează prin așa-numitele cutii S (fig. 5.9).

5.3.2. ALGORITMI PRODUS

Un algoritm produs (numit și cifru produs) reprezintă o compoziție a t funcții (cifruri) $F_1, F_2, .,. F_t$, în care fiecare F_i poate fi o substituție sau o permutare. Shannon a propus compunerea în diferite feluri a funcțiilor pentru crearea unor transformări mixte care distribuie în mod uniform mulțimea mesajelor $\mathcal{M}$ pe mulțimea tuturor criptogramelor $\mathcal{C}$.

5.3.2.1. **Rețele de subtituție-permutare (S—P)**

Aceste categorii de cifruri-produs se bazează pe rețele de cutii $S—P$, așa cum se prezintă în figura 5.10.

În exemplul considerat cifrul prelucrează blocuri de 12 biți, aplicînd substituția S_i și transpoziția P_i.

$$C = E_k(M) = S_t \circ P_{t-1} \circ \ldots \circ S_2 \circ P_1 \circ S_1(M),$$

unde fiecare funcție S_i este dependentă de o cheie K. Substituția S_i a fost divizată în 4 substituții, $S_{i1}, \ldots S_{i4}$, fiecare operînd asupra a 3 biți, pentru reducerea complexității circuitelor electronice implicate.

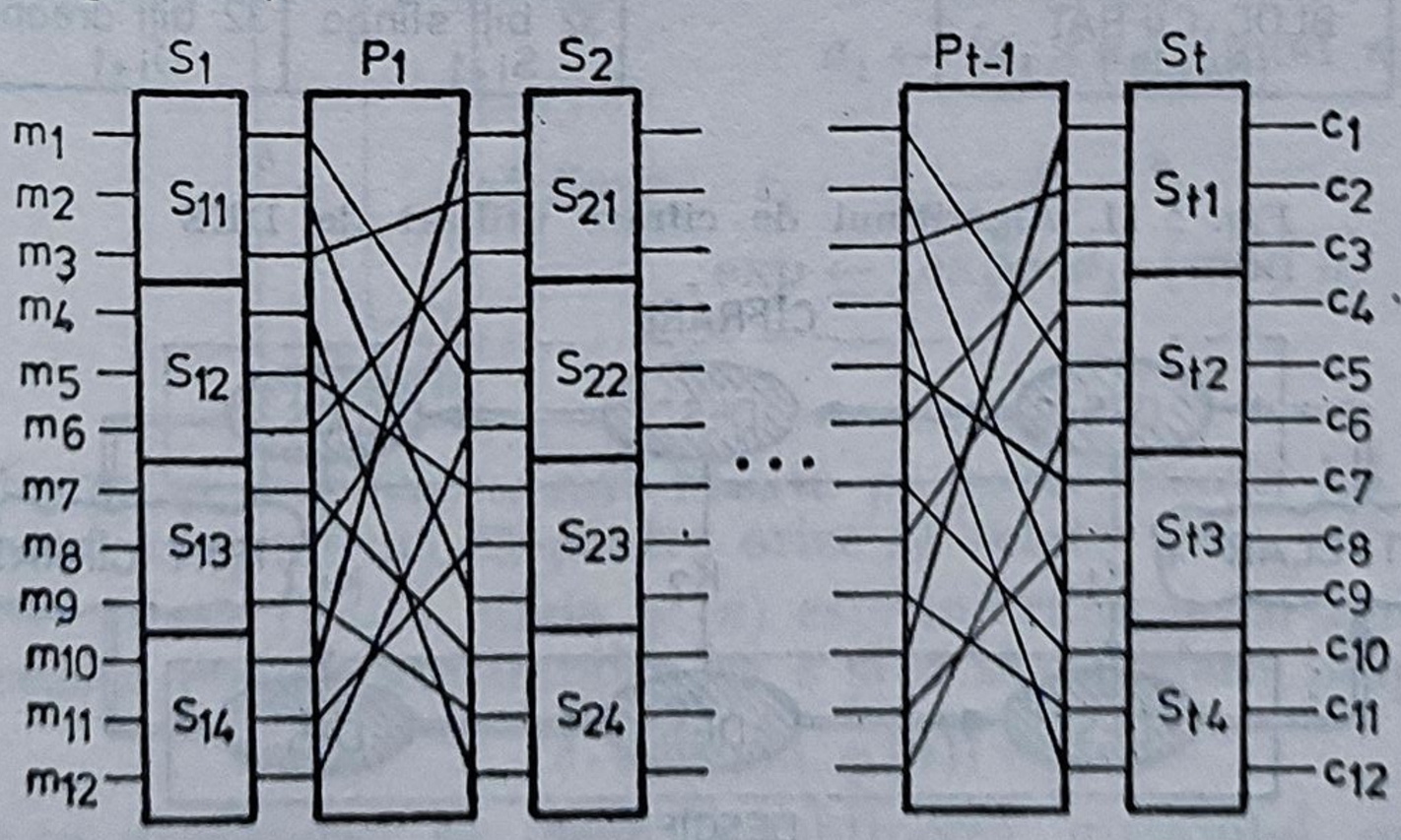


Fig. 5.10. Cifru produs

Un exemplu semnificativ de cifru bazat pe substituții-permutări îl constituie cifrul LUCIFER, proiectat de I.M.B.

5.3.2.2. Standardul DES

Standardul DES, care reprezintă dezvoltarea și perfecționarea dispozitivului LUCIFER, cifrează blocuri de mesaj de 64 de biți sub controlul unei chei de 56 de biți, extinsă cu încă 8 biți de control de paritate. Algoritmul — utilizat atît la cifrare, cît și la descifrare — este prezentat în figura 5.11. Detalii privind structura algoritmului se pot găsi în [A-1].

DES a fost implementat hardware și software. Implementarea hard a ridicat rata de cifrare la cîteva milioane de biți pe secundă. Din studiile întreprinse după publicarea standardului DES de către Diffie și Hellman precum și de către Morris, Sloane și Wyner, s-au subliniat o serie de slăbiciuni ale cifrului, dintre care două sînt considerate majore:

— *dimensiunea cheii*, considerîndu-se că cei 56 de biți nu conferă o securitate adecvată (este necesară dublarea lungimii cheii la 112 biți);

— *cutiile S* care pot avea „trape" ascunse.

Tuchman afirmă că se poate obține o creștere a securității, echivalentă dublării cheii de cifrare, utilizînd o schemă de cifrare multiplă (fig. 5.12) inventată de Matyas și ayer.

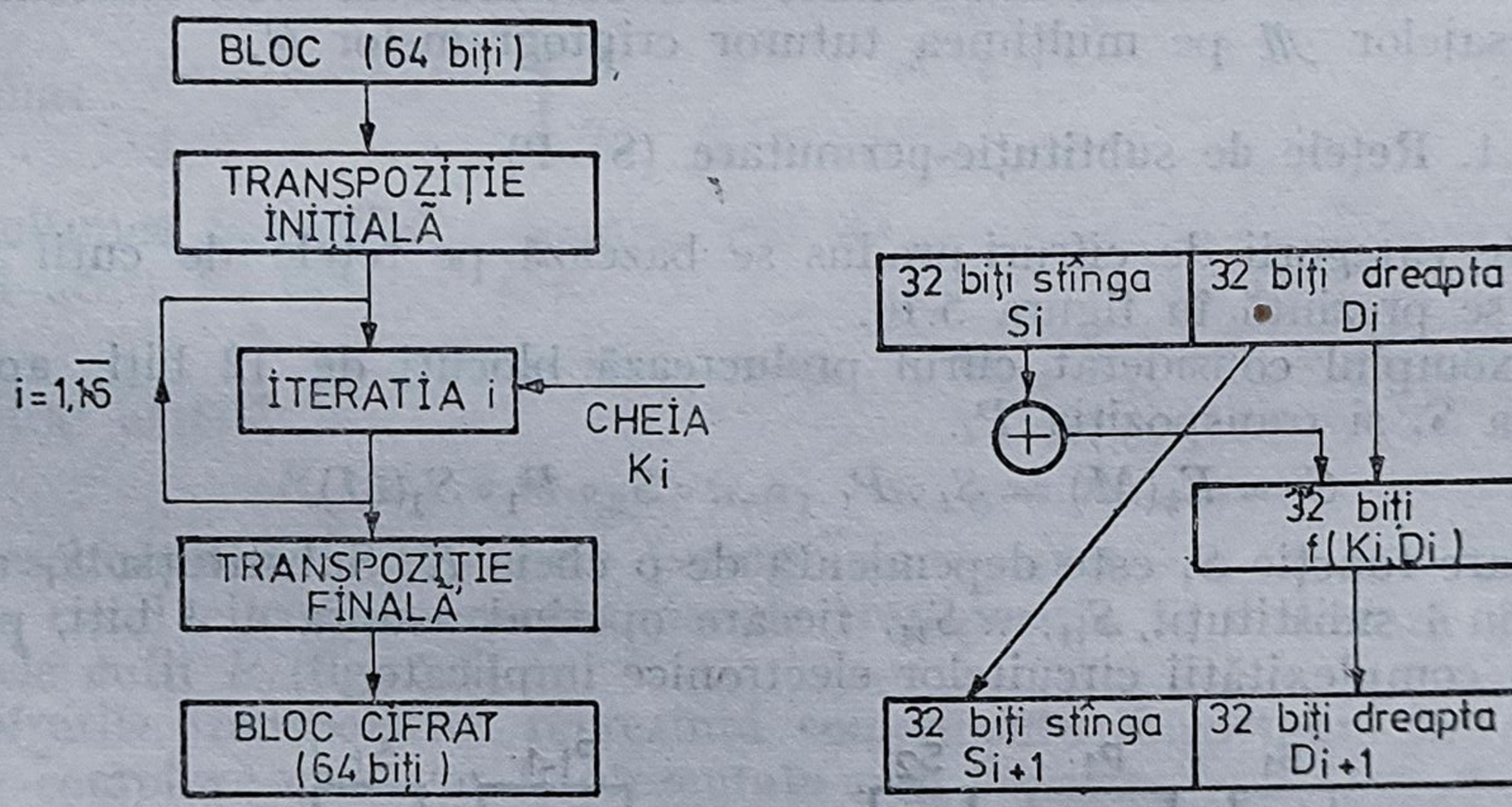


Fig. 5.11. Algoritmul de cifrare utilizat de DES

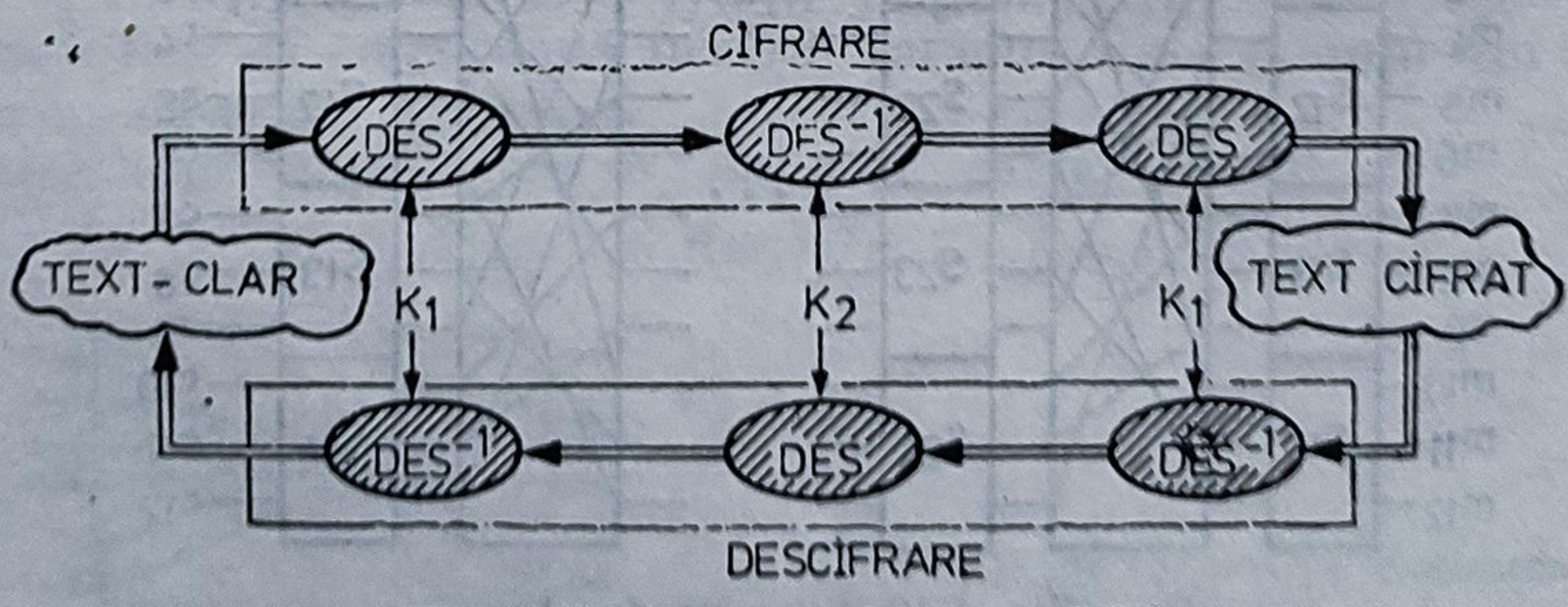


Fig. 5.12. **Cifrare multiplă cu DES**

Un mesaj este cifrat și descifrat succesiv cu chei diferite:

$$C = DES_{K1}(DES^{-1}_{K2}(DES_{K1}(M))).$$

Deși perfectibil din mai multe puncte de vedere, cifrul DES are meritul de a fi primul standard de cifrare a datelor la calculator, contribuind esențial la accelerarea cercetărilor în domeniu.

5.3.3. ALGORITMI EXPONENȚIALI

Cifrurile bazate pe algoritmi exponențiali își au originea în conceptul nou de cifrare cu cheie publică elaborat de Diffie și Hellman. În anul 1978 Pohling și Hellman publică o schemă de cifru bazată pe calculul exponențialelor într-un cîmp finit. Aproximativ în același timp, Rivest, Shamir și Adleman [R—3] comunică o schemă oarecum asemănătoare. Atît metoda Pohling-Hellman (PH), cît și metoda Rivest-Shamir-Adleman (RSA) cifrează cîte un bloc de mesaj $M \in (0, n-1)$, calculînd exponențialele:

$$C = M^e \pmod{n},$$

unde e și n reprezintă cheia transformării de cifrare. Descifrarea se face prin aceeași operație, utilizînd un exponent diferit, d, drept cheie:

$$M = C^a \pmod{n}.$$

Funcția exponențială poate fi implementată prin următorul algoritm rapid de calcul care calculează $\exp = a^z \pmod{n}$:

```
—funcția  exp (a, z, n) este
    atribuie  a1 ← a
              z1 ← z
              exp ← 1
    —cît timp z1 ≠ 0 execută
        —cît timp z1 (mod 2) = 0 execută
            atribuie z1 ← z1/2
                     a1 ← (a1 × a1) (mod n)
        —□
        atribuie  z1 ← z1 — 1
                  exp ← (exp × a1) (mod n)
    —□
—sfîrșit.
```

Cifrarea și descifrarea sînt bazate pe generalizarea lui Euler a teoremei lui Fermat, care afirmă că pentru orice M relativ prim cu n,

$M^{\varphi(n)} \pmod{1} = 1$, unde $\varphi(n)$ este indicatorul lui Euler.

Această proprietate implică ca e și d să satisfacă proprietatea:

$$e \cdot d \pmod{\varphi(n)} = 1,$$

pentru ca descifrarea să ducă la obținerea mesajului original. Deoarece $(M^a \bmod n)^e \bmod n = M^{ae} \bmod n = M$, adică algoritmul este simetric, el poate

fi folosit atît pentru cifrare cît și pentru autentificare. Fiind dat $\varphi(n)$, este ușor să se genereze perechea (e, d) care să satisfacă proprietatea cerută. Tocmai abilitatea cu care se ascunde $\varphi(n)$ deosebește schema RSA de PH.

• *Metoda PH*

În metoda PH modulul este ales ca fiind un număr prim p mare. Funcțiile de cifrare/descifrare sînt:

$$C := M^e \bmod p;$$
$$M = C^d \bmod p,$$

toate calculele făcîndu-se în cîmpul Galois GF (p). Deoarece p este număr prim, $\varphi(p) = p - 1$, deci este derivat în mod trivial din p. Ca urmare, metoda PH poate fi folosită doar într-un criptosistem convențional, atunci cînd se țin secrete atît e, cît și d. Se recomandă un $p = 2p' + 1$, unde p' este un număr prim mare.

Securitatea schemei rezidă în complexitatea calculării unor logaritmi discreți în cîmpul GF (p), criptoanalistul putînd calcula pe e (și pe d) din perechea (M, C):

$$e = \log_M C \text{ în cîmpul GF } (p).$$

Cel mai rapid algoritm, publicat de Adleman, pentru calculul unui logaritm în GF (p) cere un număr de pași egal cu:

$$T = \exp(\text{sqrt}(\ln(p)\ln(\ln(p)))).$$

Dacă p are o lungime de 200 de biți, $T = 2{,}7 \times 10^{11}$, ceea ce înseamnă (la un sistem cu 1 pas/μs) un calcul de aproximativ trei zile, iar la un p de 664 biți, $T = 1{,}2 \times 10^{23}$, ceea ce cere aproximativ 10^{12} zile, cu totul prohibitiv.

• *METODA RSA*

În cadrul acestei metode modulul n este obținut prin produsul a două numere prime mari:

$$n := p \cdot q,$$

astfel încît $\varphi(n) = (p-1)(q-1)$, devine mult mai greu de determinat, iar schema poate fi folosită cu succes într-un criptosistem cu chei publice; se vor face publice e și n, iar d va fi ținut secret. În privința metodei se recomandă alegerea unui d relativ prim cu $\varphi(n)$ în intervalul $[\max(p, q) + 1, n - 1]$. În acest caz se va calcula astfel:

$$e = \text{inv}(d, \varphi(n)),$$

putîndu-se utiliza o versiune extinsă a algoritmului lui Euclid.

Securitatea metodei depinde de dificultatea factorizării lui n în p și q. Cel mai rapid algoritm publicat de Schroeppel, cere un număr de pași egal cu:

$$T = \exp(\text{sqrt}(\ln(n)\ln(\ln(n))))$$

același cu cel al calculului logaritmului în cîmpul GF (n).

Rivert, Shamir și Adleman sugerează utilizarea unor numere prime p și q a 100 de biți, adică a unui n de 200 de biți, ceea ce cere pentru factorizare mai multe bilioane de ani.

Pentru o mai bună protecție împotriva factorizării trebuie luate și unele măsuri suplimentare în selectarea lui p și q:

— p și q să aibă lungimi care să difere cu cîțiva biți;

— atît $(p-1)$, cît și $(q-1)$ să conțină un număr mare de factori primi;

— c.m.m.d.c. $(p-1)$, $(q-1)$ trebuie să fie mic.

În proiectarea schemei este bine să se plece de la un număr prim p' generat aleator, astfel ca $p'-1$ să aibă un număr mare de factori primi. Apoi se va genera numărul prim p: $p = i \cdot p' + 1$, $i = 2, 4, 6, \ldots$, pînă cînd p este prim.

Deoarece cifrarea și descifrarea sînt funcții mutual inverse, metoda RSA poate fi utilizată atît la secretizare, cît și la autentificare. Fiecare utilizator A obține modulul n_A și exponenții e_A și $d_A \cdot A$ va înregistra într-un fișier public cheia publică (n_A, e_A), în timp ce va ține pe d_A secretă.

Un alt utilizator B va putea emite un mesaj secret M utilizînd transformarea de cifrare publică a lui A:

$$E_A(M) = M^{e_A} \bmod n_A.$$

La recepție A va obține mesajul în clar:

$$D_A(E_A(M)) = M^{e_A \cdot d_A} \bmod n_A = M.$$

Utilizatorul A va putea semna un mesaj M către B calculînd:

$$D_A(M) = M^{d_A} \bmod n_A,$$

iar B va autentifica acest mesaj, utilizînd cheia publică a lui A:

$$E_A(D_A(M)) = M^{d_A e_A} \bmod n_A = M.$$

O dificultate apare atunci cînd sînt cerute atît protecția cît și autentificarea, deoarece este necesar să se aplice transformări succesive cu module diferite. De exemplu, pentru ca A să transmită la B un mesaj semnat și secret, A va calcula:

$$C = E_B(D_A(M)).$$

Dacă $n_A > n_B$ blocul $D_A(M)$ nu mai aparține mulțimii $[0, n_B - 1]$ corespunzătoare lui E_B. Reducerea lui $D_A(M)$ mod n_B nu rezolvă problema nemaiputîndu-se apoi obține mesajul original.

Soluția constă în utilizarea unui prag h (de exemplu $h = 10^{99}$) astfel, încît fiecare utilizator să aibă două perechi de transformări: (E_{A_1}, D_{A_1}) pentru semnătură și (E_{A_2}, D_{A_2}) pentru protecție, respectînd condiția:

$$n_{A_1} < h < n_{A_2}.$$

În acest caz la transmiterea unui mesaj semnat și secretizat la B, A va calcula:

$$C = E_{B_2}(D_{A_1}(M));$$

deoarece $n_{A_1} < h < n_{B_2}$, problema este rezolvată.

Utilizatorul B va obține pe M verificînd și semnătura:

$$\begin{aligned} E_{A_1}(D_{B_2}(C)) &= E_{A_1}(D_{B_2}(E_{B_2}(D_{A_1}(M)))) \\ &= E_{A_1}(D_{A_1}(M)) \\ &= M. \end{aligned}$$

O altă soluție în situația în care $C = E_B(D_A(M))$ nu este calculabilă, deoarece $n_A > n_B$ se indică de Konfelder. El recomandă să se calculeze $C' = D_A(E_B(M))$, observînd că aceasta este calculabilă. Utilizatorul, B, care cunoaște atît n_A, cît și n_B, poate obține pe M în două moduri:

— dacă $n_A < n_B$

$$E_A(D_B(C) = E_A(D_B(E_A(D_A(M))))$$
$$= E_A(D_A(M))$$
$$= M.$$

— dacă $n_A > n_B$

$$D_B(E_A(C')) = D_B(E_A(D_A(E_B(M))))$$
$$= D_B(E_B(M))$$
$$= M.$$

Cînd apare o dispută între A și B asupra autenticității semnăturii lui A un „judecător" va trebui să fie capabil să stabilească dacă M aparține lui A:

— dacă $n_A < n_B$, B va aplica transformarea sa secretă asupra lui C și va prezenta „judecătorului" $X = D_B(C)$ și pe M. „Judecătorul" va calcula $M' = E_A(X)$, folosind transformarea publică a lui A și va compara dacă $M' = M$;

— cînd $n_A > n_B$, B se va prezenta la „judecător" cu C' și M. „Judecătorul" va calcula:

$X = E_B(M)$;

$X' = E_A(C') = E_A(D_A(E_B(M)))$ și va verifica dacă

$X = X'$.

Se observă că protocolul nu cere ca A sau B să dea „judecătorului" cheile (transformările lor secrete).

Pentru reducerea dimensiunii semnăturii se sugerează comprimarea mesajului printr-o funcție de „dispersie" și apoi semnarea rezultatului.

5.3.4. ALGORITMI DE TIP RUCSAC

În cadrul acestui paragraf se vor descrie trei metode de cifrare cu cheie publică bazate pe cunoscuta problemă a rucsacului (NP-completă). Problema rucsacului constă în a determina într-o mulțime de numere o submulțime de o sumă dată. Primele două scheme care vor fi prezentate pot fi utilizate în scopul protecției, dar nu și în cel al autentificării. Motivul constă în faptul că transformările de cifrare aplicate pe spațiul mesajelor și cu valori în același spațiu nu sînt surjective. Ca urmare, nu este posibil să se ia un mesaj oarecare și să se semneze, aplicîndu-se funcția de descifrare. Cea de a treia metodă este folosită doar pentru autentificare. Algoritmul de descifrare poate fi aplicat oricărui mesaj, însă cel de cifrare numai unui subspațiu al spațiului mesajelor.

O altă soluție în situația în care $C = E_B(D_A(M))$ nu este calculabilă, deoarece $n_A > n_B$ se indică de Konfelder. El recomandă să se calculeze $C' = D_A(E_B(M))$, observînd că aceasta este calculabilă. Utilizatorul, B, care cunoaşte atît n_A, cît şi n_B, poate obține pe M în două moduri:

— dacă $n_A < n_B$

$$E_A(D_B(C) = E_A(D_B(E_A(D_A(M))))$$
$$= E_A(D_A(M))$$
$$= M.$$

— dacă $n_A > n_B$

$$D_B(E_A(C')) = D_B(E_A(D_A(E_B(M))))$$
$$= D_B(E_B(M))$$
$$= M.$$

Cînd apare o dispută între A şi B asupra autenticității semnăturii lui A un „judecător" va trebui să fie capabil să stabilească dacă M aparține lui A:

— dacă $n_A < n_B$, B va aplica transformarea sa secretă asupra lui C şi va prezenta „judecătorului" $X = D_B(C)$ şi pe M. „Judecătorul" va calcula $M' = E_A(X)$, folosind transformarea publică a lui A şi va compara dacă $M' = M$;

— cînd $n_A > n_B$, B se va prezenta la „judecător" cu C' şi M. „Judecătorul" va calcula:

$X = E_B(M)$;

$X' = E_A(C') = E_A(D_A(E_B(M)))$ şi va verifica dacă

$X = X'$.

Se observă că protocolul nu cere ca A sau B să dea „judecătorului" cheile (transformările lor secrete).

Pentru reducerea dimensiunii semnăturii se sugerează comprimarea mesajului printr-o funcție de „dispersie" şi apoi semnarea rezultatului.

5.3.4. ALGORITMI DE TIP RUCSAC

În cadrul acestui paragraf se vor descrie trei metode de cifrare cu cheie publică bazate pe cunoscuta problemă a rucsacului (NP-completă). Problema rucsacului constă în a determina într-o mulțime de numere o submulțime de o sumă dată. Primele două scheme care vor fi prezentate pot fi utilizate în scopul protecției, dar nu şi în cel al autentificării. Motivul constă în faptul că transformările de cifrare aplicate pe spațiul mesajelor şi cu valori în același spațiu nu sînt surjective. Ca urmare, nu este posibil să se ia un mesaj oarecare şi să se semneze, aplicîndu-se funcția de descifrare. Cea de a treia metodă este folosită doar pentru autentificare. Algoritmul de descifrare poate fi aplicat oricărui mesaj, însă cel de cifrare numai unui subspațiu al spațiului mesajelor.

• **Metoda Merkle-Hellman** (MH)

Merkle și Hellman [M—3] propun o metodă a cărei securitate depinde de dificultatea rezolvării următoarei probleme:

Fiind dat un întreg pozitiv C și un vector $A = (a_1, a_2, ..., a_n)$ de întregi pozitivi, să se găsească un subset al lui A a cărui sumă să fie C. Cu alte cuvinte este necesar să se determine un vector binar $M = (m_1, ..., m_n)$, astfel ca $C = A \cdot M$ sau:

$$C = \sum_{i=1}^{n} a_i \cdot m_i.$$

Această problemă este NP completă. Cel mai bun algoritm cunoscut pentru rezolvarea ei — în cazul în care dimensiunea „rucsacului" este n— cere $0\ (2^{n/2})$ timp și $0\ (2^{n/2})$ memorie. Există totuși o clasă specială de probleme de asemenea tip, numită cu „rucsac simplu" ce pot fi rezolvate într-un timp linear. Într-un rucsac simplu elementele a_i $(i = 1, ..., n)$ sînt în relația de dominanță, adică:

$$a_i > \sum_{j=1}^{i-1} a_j.$$

Relația de dominanță simplifică soluția „rucsacului", ea putîndu-se face după următorul algoritm:

```
—procedura RUCSAC-SIMPLU (C, A; M) este
  —pentru I = N, 1 execută
    —dacă C ⩾ a_i atunci  atribuie m_i ← 1
    -□          altfel    atribuie m_i ← 0
      atribuie C ← C — a_i · m_i
  -□ —dacă C = 0 atunci „soluția este în M"
    -□           altfel „nu există soluție"
—sfîrșit.
```

În proiectarea unui astfel de criptosistem trebuie însă convertit „rucsacul simplu" într-un „rucsac cu trapă", care este greu de rezolvat. Mai întîi se selectează un vector „rucsac" simplu" $A' = (a'_1, a'_2, ..., a'_n)$. Acesta permite o soluție simplă a problemei, $C' = A' \cdot M$. Apoi se alege un întreg n, astfel ca:

$$n > 2 \cdot a'_n > \sum_{i=1}^{n} a'_i.$$

În continuare se alege un alt întreg W, astfel ca c.m.m.d.c. $(n, W) = 1$ și se calculează W^{-1}, inversa lui W mod n. În final vectorul A' este transformat într-un vector „rucsac greu" $A = W \cdot A' \bmod n$:

$$a_i = W \cdot a'_i \bmod n.$$

Acum rezolvarea problemei $C = A \cdot M$ este dificilă, dacă nu se dispune de o informație „trapă" (W^{-1} și n), cu ajutorul căreia calculul se simplifică:

$$\begin{aligned} C' &= W^{-1} \cdot C \quad \bmod n; \\ &= W^{-1} \cdot A \cdot M \quad \bmod n; \\ &= W^{-1}(WA')\, M \quad \bmod n; \\ &= A'M \quad \bmod n; \\ &= A'M. \end{aligned}$$

Transformarea de cifrare E_A (publică) folosește cheia publică care este vectorul, „rucsac greu" A. Se obține criptograma:

$$C = E_A(M) = A \cdot M.$$

Descifrarea D_A folosește cheia secretă (A', n, W^{-1}), calculînd pe baza funcției „rucsac simplu":

$$D_A(C) = \text{rucsac-simplu } (W^{-1} \cdot C \bmod n, A') = M.$$

Inițial Merkle și Hellman au sugerat alegerea unui $n = 100$ sau mai mare în proiectarea unui criptosistem bazat pe acest algoritm. Schroeppel și Shamir au dezvoltat însă un algoritm care rezolvă un astfel de rucsac într-un timp $T = 0\ (2^{n/2})$ și cu o memorie $S = 0\ (2^{n/2})$. Pentru $n = 100$, $2^{50} \approx 10^{15}$, ceea ce înseamnă că un singur procesor poate găsi soluția în 11 574 de zile pe cînd 1 000 de procesoare determină soluția în 12 zile. Însă pentru $n = 200$ algoritmul este computațional inatacabil.

- **Metoda Graham-Șhamir**

Graham și Shamir au găsit independent o cale pentru ascunderea proprietății de dominanță a „rucsacului cu trapă". Conform lor, vectorul cu trapă devine $A' = (a'_1, \ldots a'_n)$, cu proprietatea că fiecare a_i are următoarea reprezentare binară:

$$a_i = (R_i I_i S_i)$$

unde R_i și S_i sînt vectori binari aleatorii și lungi, iar I_i este un șir de biți de lungime n, cu un bit, al i-lea, pe 1 și restul de $(n-1)$ pe 0. Fiecare șir aleatoriu S_i are $\log_2 n$ de 0 în pozițiile cele mai semnificative, în așa fel încît însumarea să nu producă depășire în domeniul lui I_i. Suma $C' = A'M$ are reprezentarea binară:

$$C' = (R\, M\, S),$$

unde

$$R = \sum_{i=1}^{n} R_i \cdot m_i \quad \text{și} \quad S = \sum_{i=1}^{n} S_i \cdot m_i.$$

Vom observa că vectorul de șiruri de biți

$$((I_n\, S_n) \ldots (I_1, S_1)),$$

adică elementele a'_i în ordine inversă, alcătuiesc un rucsac simplu. Elementele R_i sînt adunate pentru a ascunde această proprietate.

Vectorul rucsac A' va fi convertit într-unul dificil de rezolvat A, într-un mod similar cu cel indicat la schema MH: se aleg n și W și cu ajutorul lor se calculează $A = W \cdot A'$ mod n. Cifrarea va fi deci:

$$C = E_A(M) = A \cdot M,$$

iar descifrarea constă din calcularea lui:

$$C' = W^{-1} \cdot C \text{ mod } n$$

și apoi extragerea din C' a biților lui M.

Se apreciază că această metodă este mai sigură, rapidă și simplu de implementat decît schema MH.

• Metoda Shamir

Algoritmul publicat de Shamir permite realizarea doar a prcesului de autentificare. Vectorul „rucsac cu trapă" proiectat conform metodei propuse permite implementarea semnăturii digitale, nu însă și a protecției. Soluția se bazează pe următoarea problemă NP-completă:

Fiind dați întregii n, M și vectorul $A = (a_1 \ldots a_{2K})$ să se găsească $C = (c_1, \ldots, c_{2K})$, astfel ca $M = C \cdot A$ mod n. Aceasta înseamnă că:

$$M = \sum_{i=1}^{2K} c_i a_i \text{ mod } n,$$

unde fiecare $c_i \in [0, \log n]$. În schema n este un număr prim de K biți ($K = 100$ de exemplu). Perechea (A, n) alcătuiește cheia publică, M este mesajul $M \in [0, (n-1)]$ și C reprezintă semnătura lui M. Receptorul perechii (M, C) poate valida semnătura, calculînd:

$$E_A(C) = C \cdot A \cdot \text{ mod } n$$

și verificînd dacă rezultatul coincide cu M.

Receptorul însă nu va putea falsifica semnătura unui alt mesaj M'. Semnatarul dispune de o informație-trapă pentru generarea semnăturii $C = D_A(M)$. Această informație este reprezentată de matricea liniară H de $K \times 2K$, ale cărei valori sînt alese în mod aleator. Vectorul A este astfel construit, încît să satisfacă următorul sistem de ecuații liniare:

$$\begin{bmatrix} h_{1,1} \ldots h_{1,2K} \\ h_{2,1} \ldots h_{2,2K} \\ \vdots \\ h_{K,1} \ldots h_{K,2K} \end{bmatrix} \cdot \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_{2K} \end{bmatrix} = \begin{bmatrix} 2^0 \\ 2^1 \\ \vdots \\ 2^{K-1} \end{bmatrix} \text{ mod } n$$

adică:

$$\sum_{j=1}^{2K} h_{i,j} \cdot a_j = 2^{i-1} \text{ mod } n, \quad i = 1, \ldots, K.$$

Deoarece sînt K ecuații cu $2K$ necunoscute, K valori ale lui A pot fi alese aleator, iar restul se obțin din rezolvarea sistemului de ecuații indicat.

Fie $M = (m_K, m_{K-1}, ..., m_1)$ mesajul clar și $\overline{M} = (m_1, ... m_{K-1}, m_K)$ obținut prin inversarea ordinii biților lui M. Semnătura se va obține calculînd:

$$C = D_A(M) = \overline{M} \cdot H,$$

adică:

$$c_j = \sum_{i=1}^{K} m_i \cdot h_{i,j} \quad (1 \leqslant j \leqslant 2K)$$

Validarea semnăturii se va face astfel:

$$\begin{aligned} E_A(C) = C \cdot A \bmod n &= \sum_{j=1}^{2K} c_j \cdot a_j \bmod n \\ &= \sum_{j=1}^{2K} \left(\sum_{i=1}^{K} m_i \cdot h_{i,j} \right) a_j \bmod n \\ &= \sum_{i=1}^{K} m_i \left(\sum_{j=1}^{2K} h_{i,j} \cdot a_j \right) \bmod n \\ &= \sum_{i=1}^{K} m_i 2^{i-1} \bmod n \\ &= M. \end{aligned}$$

În vederea împiedicării determinării lui H prin examinarea unor perechi (M, C), la proiectarea unui astfel de criptosistem se indică randomizarea mesajelor înainte de a fi semnate. Acest lucru se realizează cu ajutorul unui vector binar $R = (r_1, ..., r_{2K})$, calculînd:

$$M' = (M - RA) \bmod n,$$

iar apoi

$$M = (M' + RA) \bmod n.$$

Mai întîi este semnat M' obținîndu-se C', apoi semnătura lui M se determină din C':

$$C = C' + R.$$

C este semnătura validă a lui M, deoarece:

$$\begin{aligned} C \cdot A \bmod n &= (C' + R) A \bmod n \\ &= (C' \cdot A + RA) \bmod n \\ &= (M' + RA) \bmod n \\ &= M. \end{aligned}$$

Întrucît semnătura depinde de vectorul aleator R, M poate avea mai multe semnături C care să satisfacă sistemul de ecuații:

$$M = \sum_{j=1}^{2K} c_j \cdot a_j \bmod n.$$

Din acest motiv schema nu poate fi folosită pentru protecție (nu se poate aplica cu succes $D_A(E_A(M))$.

5.4. PROIECTAREA SISTEMELOR ȘI PROTOCOALELOR CRIPTOGRAFICE

Există diferite modalități de implementare și integrare a algoritmilor de cifrare prezentați anterior în sisteme și protocoale de control și protecție specifice sistemelor teleinformatice, dintre care unele vor face obiectul analizei noastre în continuare.

5.4.1. SISTEME CRIPTOGRAFICE BLOC ȘI SECVENȚIALE

Fie M un mesaj clar. Un sistem criptografic bloc divide M în blocuri succesive M_1, M_2, ... și cifrează fiecare M cu aceeași cheie K:

$$E_K(M) = E_K(M_1)\, E_K(M_2) \ldots$$

Un sistem criptografic secvențial divide mesajul M în caractere succesive sau biți m_1, m_2,... și cifrează fiecare m_i cu elementul k_i al șirului cheie $K = k_1, k_2, \ldots$:

$$E_K(M) = E_{k_1}(m_1)\, E_{k_2}(m_2) \ldots .$$

Toate cifrurile secvențiale analizate în continuare vor utiliza pentru cifrare/descifrare o operație simplă de SAU — EXCLUSIV:

$$E_{k_i}(m_i) = m_i \oplus k_i = c_i ;$$

$$D_{k_i}(c_i) = c_i \oplus k_i = (m_i \oplus k_i) \oplus k_i = m_i ,$$

unde c_i, m_i și k_i sînt caractere de 1 bit fiecare.

5.4.2. PROTECȚIA DATELOR PRIN SISTEME SECVENȚIALE SINCRONE

În cazul cifrurilor secvențiale sincrone șirul-cheie $K = k_1, k_2 \ldots$ este generat independent de caracterele mesajului. Algoritmul de generare trebuie să fie determinist, astfel încît șirul să poată fi reprodus la descifrare. Valoarea de declanșare a generatorului de cheie va fi inițializată cu valoarea I_0 (fig. 5.13). Un astfel de cifru este inatacabil din punct de vedere criptoanalitic dacă șirul-cheie este aleatoriu și de lungime egală cu lungimea textului clar. Nici un algoritm finit nu poate genera secvențe aleatoare, de aceea pentru o serie de implementări reale se utilizează generatoare de cheie pseudoaleatoare bazate pe registru de deplasare, dintre care cîteva vor fi analizate în cele ce urmează.

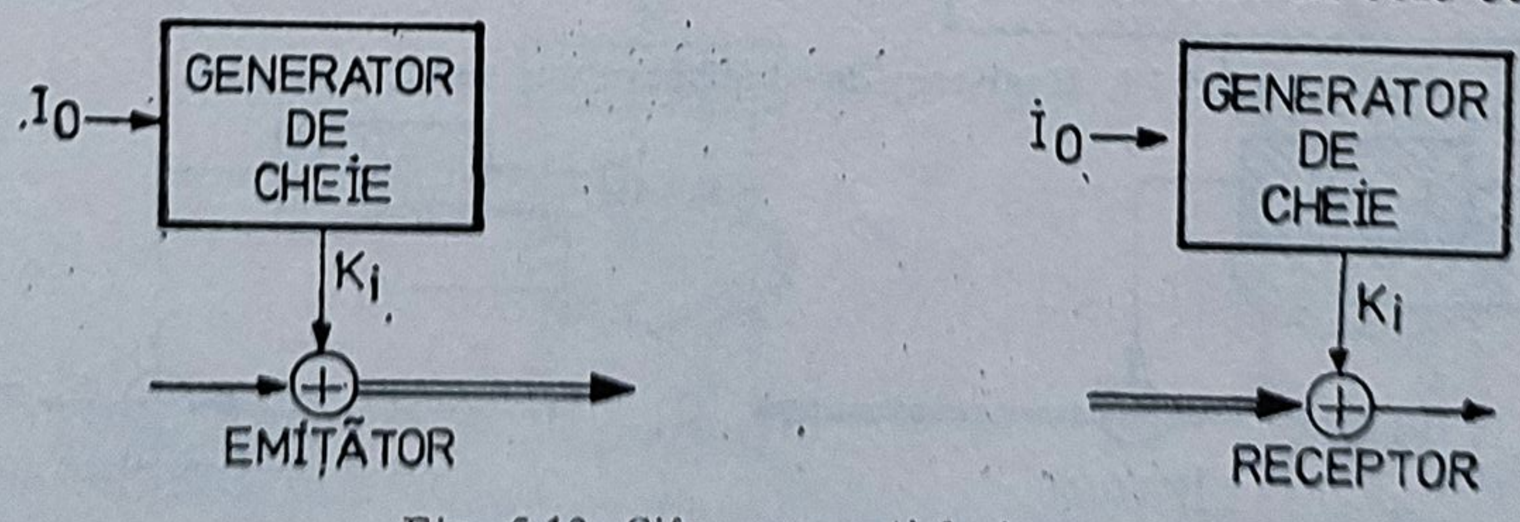


Fig. 5.13. Cifru secvențial sincron

• Registru de deplasare cu reacție liniară

Un registru de deplasare cu reacție liniară (LFSR) constă dintr-un registru $R = (r_n, r_{n-1}, ..., r_1)$ și o secvență „bandă" $T = (t_n, t_{n-1}, ..., t_1)$, unde fiecare t_i și r_i sînt biți. La fiecare pas, bitul r_1 este adăugat la șirul-cheie, biții $r_n ... r_2$ sînt deplasați cu o parte la dreapta și un nou bit, derivat din T și R este inserat în poziția r_n (fig. 5.14). Dacă $R' = (r'_n, ..., r'_1)$ este noua stare a registrului R, avem:

$$r'_i = r_{i+1}, \quad i = 1, ..., n-1;$$

$$r'_n = T \cdot R = \sum_{i=1}^{n} t_i \cdot r_i \bmod 2 = t_1 \cdot r_1 \oplus ... \oplus t_n \cdot r_n,$$

adică:

$$R' = R \cdot H \bmod 2,$$

unde H este o matrice $n \times n$:

$$H = \begin{bmatrix} t_n & t_{n-1} & ... & t_2 & t_1 \\ 1 & 0 & ... & 0 & 0 \\ 0 & 1 & ... & 0 & 0 \\ \vdots & & & & \\ 0 & 0 & ... & 1 & 0 \\ 0 & 0 & ... & 0 & 1 \end{bmatrix}.$$

Un astfel de generator pseudoaleator LFSR poate genera secvențe cu perioada $2^n - 1$. Acest lucru este realizat atunci cînd polinomul

$$T(x) = t_n x^n + ... t_1 x + 1$$

este primitiv, adică divide pe $x^{2^n-1} + 1$, dar nu și pe $x^d + 1$, pentru orice d care divide pe $2^n - 1$. În proiecarea unui LSFR se preferă polinoame primitive de forma $T(x) = x^n + x^a + 1$, deoarece selectează doar două din elementele registrului R pentru regenerarea reacției.

Cifrarea mesajelor cu ajutorul generatoarelor LFSR este ilustrată în figura 5.15.

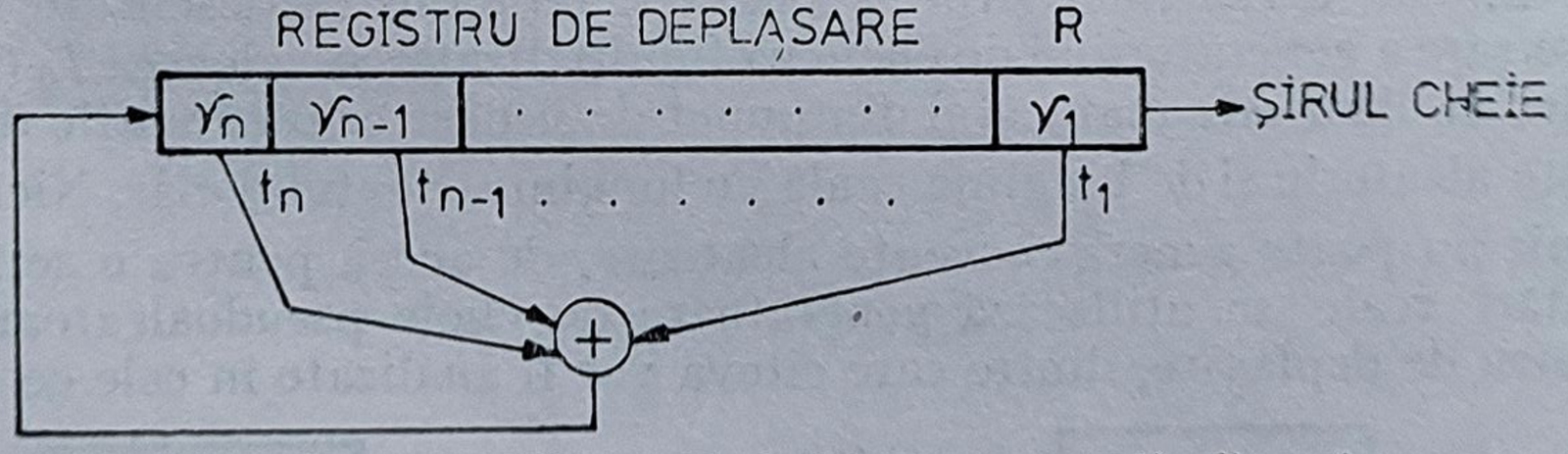
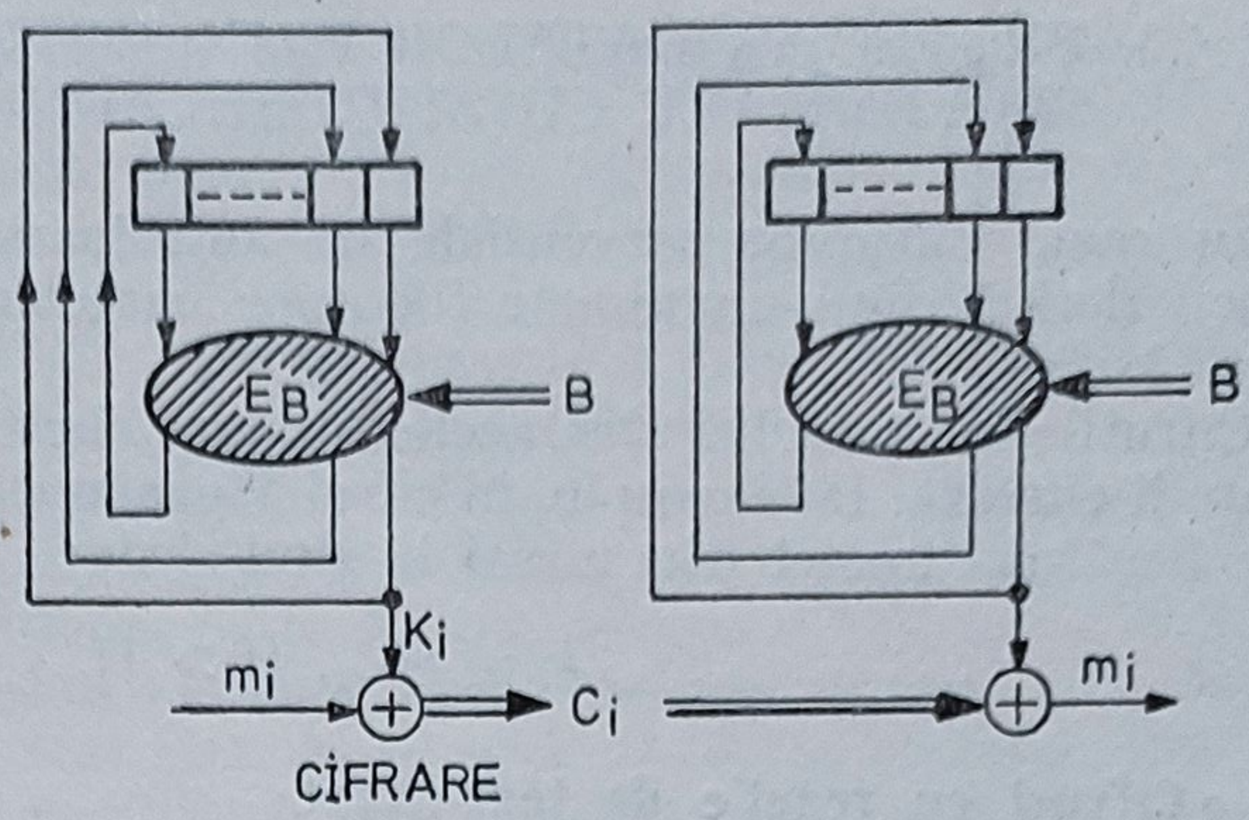


Fig. 5.14. Registru de deplasare cu reacție liniară

Fig. 5.15. Cifrare cu LFSR

Fig. 5.16. Cifru secvențial sincron cu reacție bloc



Acest gen de reacții dorește să simuleze cifrul Vernam cu o singură utilizare a cheii, transformînd o cheie compactă, I_u, într-o secvență lungă, K. Din nefericire secvența T se obține ușor la un atac cu text clar. Numărul de operații cerute pentru calculul lui T este aproximativ n^3, ceea ce presupune o zi de calcul pentru un $n = 1\,000$.

• *Generator cu reacție-bloc*

Slăbiciunea metodei LFSR este determinată de liniaritatea ecuației. O abordare superioară constă din utilizarea unei transformări neliniare. Cifrurile-bloc neliniare, precum DES, pot fi utilizate în acest scop. În figura 5.16 se prezintă metoda numită generator cu reacție-bloc. Registrul de reacție R este utilizat ca intrare a algoritmului de cifrare bloc E_B cu cheia B. În timpul celei de a i-a iterații se calculează $E_B(R)$, iar caracterul din dreapta devine cel de al i-lea caracter k_i al cheii. Întreg blocul de ieșire este trecut în R și utilizat ca intrare în iterația următoare. Vom observa că fiecare caracter k_i este format din mai mulți biți pentru a reduce numărul de cifrări cu E_B (poate fi DES, de exemplu), care consumă considerabil mai mult timp decît registrul LFSR. Un mesaj este divizat în caractere și cifrat în paralel cu generarea șirului cheie. Tehnica se mai numește cu reacție internă.

• *Generator cu contor*

Diffie și Hellman [D—8] au elaborat o cale deosebită de abordare a generării pseudoaleatoare, numită metoda contor. În loc să se recicleze ieșirea lui E_B înapoi la intrarea lui E_B se generează blocuri succesive de intrare printr-un simplu contor (fig. 5.17). În cadrul metodei este posibil să fie generat direct cel de al i-lea caracter k_i al cheii prin încărcarea contorului cu valoarea $I_0 + (i - 1)$, ceea ce conferă calități ce fac utilizabil cifrul la protecția unor fișiere cu acces direct.

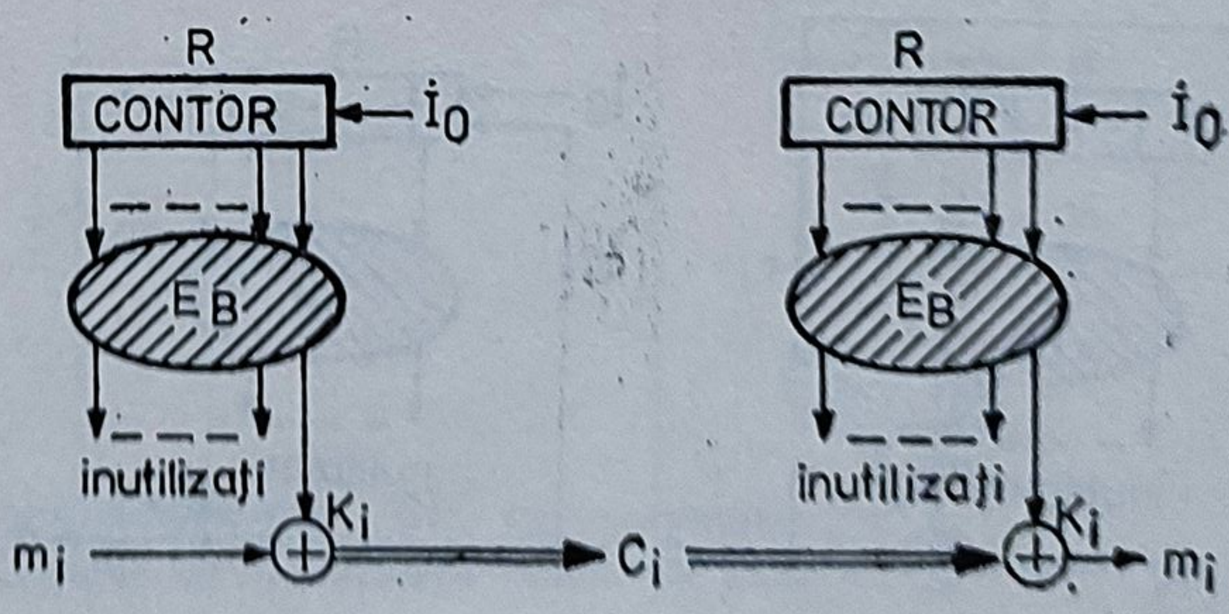


Fig. 5.17. Cifru secvențial sincron cu contor

5.4.3. PROTECȚIA DATELOR PRIN SISTEME SECVENȚIALE CU AUTOSINCRONIZARE

În cazul cifrurilor secvențiale cu autosincronizare fiecare caracter k_i al cheii derivă dintr-un număr fix de n caractere precedente ale textului cifrat.

Cifrurile cu autochei sînt acelea în care cheia este derivată din mesajul pe care îl cifrează. De exemplu, în cifrul Vigenère cheia este formată prin adăugarea fiecărui caracter din mesaj la șirul-cheie:

$$k_i = m_{i-1}, \ (i > 1).$$

- **Cifruri cu reacție de text-cifrat**

Cifrurile cu autochei sînt slabe deoarece expun cheia în șirul de caractere cifrate. Deficiența este însă ușor înlăturată dacă caracterele cifrate sînt trecute printr-un cifru-bloc neliniar (de exemplu, DES) pentru a se genera șirul-cheie. Tehnica se numește reacție cu text-cifrat, întrucît caracterele cifrate sînt cele care intră în reacție. Acest mod de lucru se mai numește cu înlănțuire, fiind aprobat de National Bureau of Standards pentru utilizarea cu DES.

În figura 5.18 se ilustrează această tehnică. Registrul de reacție R este un registru de deplasare; la fiecare pas, caracterul cifrat c_i este introdus în stînga odată cu deplasarea conținutului lui R cu o poziție la dreapta. Conținutul lui R devine intrare pentru cifrul-bloc E_B, a cărui ieșire este utilizată, parțial, pentru generarea caracterului k_i al șirului-cheie.

5.4.4. PROTECȚIA DATELOR PRIN SISTEME-BLOC

Un cifru-bloc divide mesajul clar M în n blocuri succesive, $M_1, M_2, \ldots, M_n$, cifrînd pe fiecare cu aceeași cheie K:

$$E_K(M) = E_K(M_1)\, E_K(M_2) \ldots E_K(M_n).$$

Am văzut în cadrul paragrafului anterior că o posibilitate de utilizare a cifrurilor bloc este la generarea șirului-cheie în bucla de reacție. Dar cifrurile-bloc pot fi folosite și ca sisteme criptografice de sine stătătoare, cu unele avantaje dar și o serie de inconveniente. Dintre avantaje putem aminti: la această categorie de cifruri o eroare de transmisie nu afectează decît blocul la care

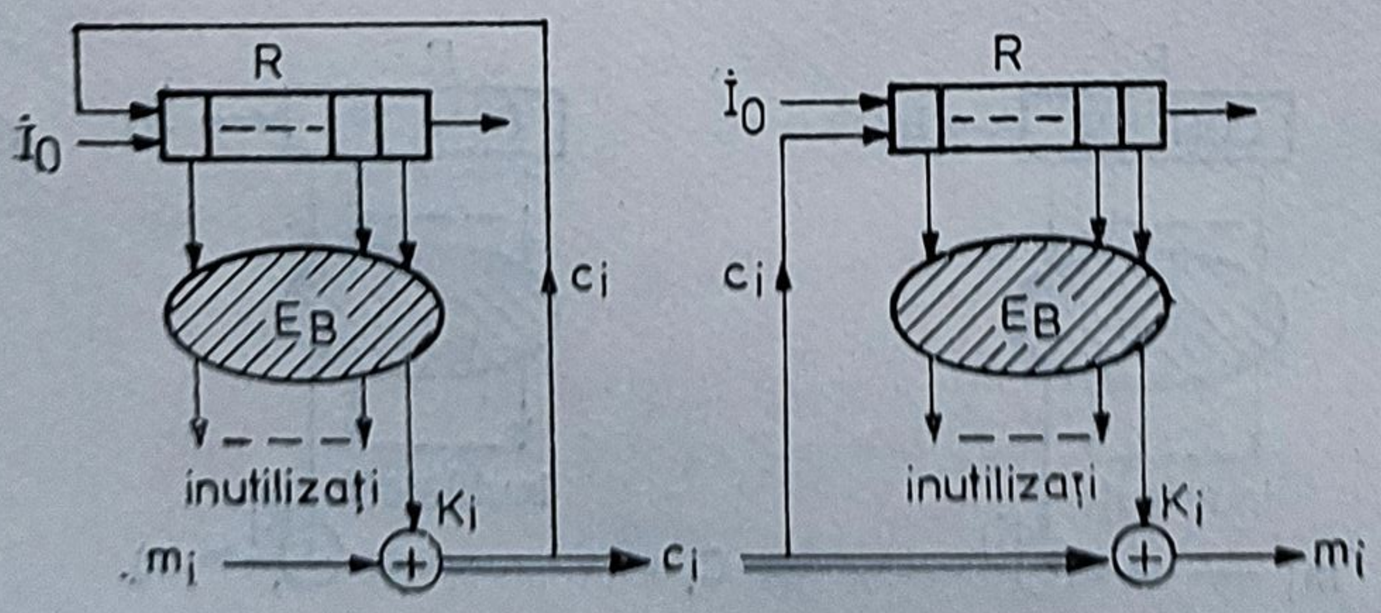


Fig. 5.18. Cifru secvențial cu autosincronizare cu reacție de text cifrat

s-a produs, timpul de cifrare este mai scurt, pot fi folosite cu succes la cifrarea unor informații în fișiere și baze de date. Dintre dezavantaje subliniem următoarele: cifrarea-bloc este susceptibilă la criptoanaliză, deoarece blocuri identice de text-clar dau blocuri identice de text cifrat; cifrarea bloc este vulnerabilă la înscrieri sau ștergeri de blocuri întrucît schimbările în șirul-mesaj nu sînt sesizate; în baza de date pot fi copiate anumite cîmpuri cifrate de la o entitate la alta, ceea ce poate crea dificultăți majore.

În cele ce urmează se va descrie o cale pentru a face cifrurile-bloc mai rezistente la atac. Feistel a arătat că cifrurile-bloc pot fi făcute mai rezistente la criptoanaliză și la substituirea de text cifrat (inclusiv recopierea unor porțiuni din textul cifrat) utilizînd o tehnică numită *înlănțuirea blocurilor*. Înainte de cifrarea fiecărui bloc de text clar, M_i, o serie de biți ai blocului cifrat precedent C_i se vor insera în pozițiile binare neutilizate ale blocului M_i. Kent a propus o abordare similară, utilizînd o secvență de numere. Ambele strategii protejează împotriva inserării, ștergerii și modificării, dar se reduce numărul de biți utili pe bloc.

Remediul constă în utilizarea unei alte tehnici, numită *înlănțuirea blocurilor cifrate*, recomandată să fie folosită cu DES. Metoda este similară cifrării cu reacție, cu deosebirea că întregul bloc de text cifrat este introdus într-un registru pentru a fi însumat modulo 2 cu următorul bloc de text clar (fig. 5.19). Cifrarea se face astfel:

$$C_i = E_K(M_i \oplus C_{i-1}), \quad C_0 = I_0.$$

Descifrarea se realizează calculînd:

$$\begin{aligned} D_K(C_i) \oplus C_{i-1} &= D_K(E_K(M_i \oplus C_{i-1})) \oplus C_{i-1} \\ &= (M_i \oplus C_{i-1}) \oplus C_{i-1} \\ &= M_i. \end{aligned}$$

Deoarece C_i se obține din M_i și C_{i-1}, orice eroare de transmisie afectează două blocuri succesive. În același timp, datorită faptului că fiecare C_i este funcțional dependent de toate blocurile cifrate precedente, proprietățile statistice ale textului clar sînt difuzate în întreg textul cifrat, făcînd criptoanaliza mult mai dificilă. Această metodă este utilizată în sistemul IPS (Information Protection System), un ansamblu de programe criptografice proiectat de I.B.M.

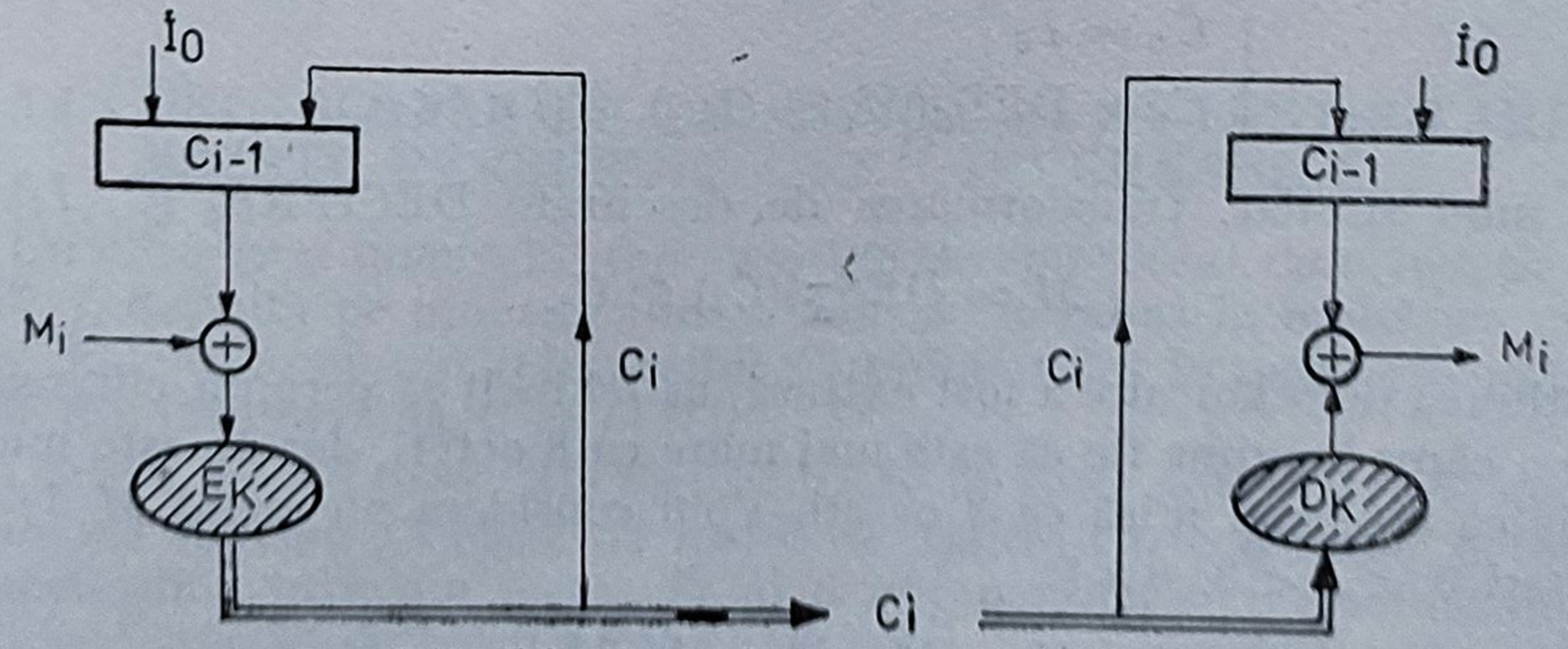


Fig. 5.19. Cifrarea cu înlănțuirea blocurilor cifrate

• **Programele criptografice IPS**

Sistemul IPS a fost dezvoltat pentru protecția criptografică locală sau în rețea a datelor confidențiale la Centrul de cercetări Thomas Watson al companiei IBM. Este format dintr-un ansamblu de programe criptografice care folosesc un cifru-bloc, a cărei transformare de bază o constituie algoritmul DES. IPS augmentează însă acest algoritm de cifrare, permițînd cifrarea de date de orice lungime prin utilizarea *înlănțuirii*.

La baza proiectării sistemului IPS au stat următoarele principii:

— nici o cheie sau variabilă echivalentă cheii nu este rezidentă în sistemul de calcul; utilizatorii sînt aceia care răspund de gestiunea cheilor, ei le introduc în sistem în momentul cifrării sau descifrării datelor;

— IPS oferă mai multe posibilități pentru utilizatori în ceea ce privește introducerea cheilor; aceștia pot introduce fie direct cei 56 de biți ai cheii necesară la DES, fie un șir lung de caractere ce va fi convertit intern într-o cheie conform cu cerințele DES;

— IPS este astfel proiectat încît să realizeze cifrarea unor date de orice lungime, chiar mai mică de 8 octeți; de asemenea, lungimea mesajului se conservă prin cifrare;

— prin utilizarea înlănțuirii, IPS cifrează zone de mesaje identice în zone de text cifrat diferite;

— IPS furnizează facilități de utilizare a unor proceduri criptografice din programe scrise în FORTRAN, PL/I, ASSEMBLER;

— fișierele cifrate cu IPS conțin un header care memorează informații privind tipul cifrării utilizate, versiunea IPS folosită, informația de înlănțuire, un cîmp de verificare a cheii pe care o furnizează utilizatorul de descifrare.

IPS a introdus noțiunea de *înlănțuire*, care are rolul de a masca în cifru repetările din textul clar. Înlănțuirea este realizată la două niveluri: bloc și articole.

Pentru a ilustra modul de realizare la nivel de bloc să considerăm următorul exemplu: Fie $M = M_1, M_2, ..., M_n$ un mesaj (articol) format din n blocuri de lungime de 8 octeți, o cheie K și o valoare inițială de înlănțuire I_0 de 8 octeți. Vom defini transformarea de cifrare înlănțuită a unui articol, $\text{CIFRA}_K[M, I_0]$, astfel:

$$\begin{cases} C_0 = I_0 \\ C_i = \text{DES}_K(M_i \oplus C_{i-1}), \quad i \leqslant i \leqslant n. \end{cases}$$

În mod similar, transformarea de descifrare $\text{DECIFRA}_k [C, I_0]$ este:

$$M = DES_K^{-1}(C_i) \oplus C_{i-1}.$$

Noțiunea de înlănțuire a fost extinsă, astfel încît să permită cifrarea unor articole a căror lungime fie că este mai mare ca 8 octeți, dar nu este multiplu de 8, fie că este mai mică ca 8 octeți. Vom considera un mesaj M de $8n + s$ octeți $0 < s < 8$, $n \geqslant 1$:

$$M = M_1, M_2 \ldots M_n M_{n+1},$$

unde M_{n+1} este blocul scurt de lungime s. Funcția de cifrare înlănțuită a unui articol, $\text{CIFRA}_K[M, I_0]$, devine:

$$\begin{cases} C_0 = I_0 \\ C_i = \text{DES}_K(M_i \oplus (C_{i-1}), \quad 1 \leqslant i \leqslant n, \\ C_{n+1} = M_{n+1} \oplus \text{STÎNGA}_s[\text{DES}_K(M_n)], \end{cases}$$

unde STÎNGA_s semnifică cei s octeți din stînga blocului de 8 octeți.

Funcția de descifrare înlănțuită DECIFRA este definită ca:

$$\begin{cases} M_i = \text{DES}_K^{-1}(C_i) \oplus C_{i-1}, \quad 1 \leqslant i \leqslant n \\ M_{n+1} = C_{n+1} \oplus \text{STÎNGA}_s[\text{DES}_K(M_n)]. \end{cases}$$

În cazul articolelor scurte (sub 8 octeți) cifrarea se definește astfel:

$$C = M \oplus \text{STÎNGA}_s[\text{DES}_K(I_0)].$$

Înlănțuirea la nivelul articolelor permite cifrarea diferită a unor articole care încep cu fragmente identice de text clar. Fie F un fișier format dintr-o secvență de r articole:

$$F: M^{(1)}M^{(2)} \ldots M^{(r)}.$$

Vom nota cu I_0 valoarea inițială de înlănțuire. În urma cifrării fișierului devine:

$$F: C^{(1)}C^{(2)} \ldots C^{(r)}$$

Pentru un articol i vom conveni ca $II_0^{(i)}$ să reprezinte valoarea de înlănțuire la intrare, iar $OI_0^{(i)}$ — valoarea de înlănțuire la ieșire.
Funcția de cifrare a fișierului, $\text{CIFRA}_K[F, I_0]$, se definește astfel:

$$\begin{cases} II_0^{(1)} = I_0 \\ II_0^{(i)} = OI_0^{(i-1)}, \quad 1 < i \leqslant r \\ C^{(i)} = \text{CIFRA}_K[M^{(i)}, II_0^{(i)}] \quad 1 \leqslant i \leqslant r \\ OI_0^{(i)} = \text{DREAPTA}_8[II_0^{(i)}, C^{(i)}] \quad 1 \leqslant i \leqslant \text{r}. \end{cases}$$

După cum se observă primul articol $\text{M}^{(1)}$ este cifrat prin DES în $C^{(1)}$ utilizînd valoarea de înlănțuire I_0, iar cel de al i-lea articol $M^{(i)}$ va fi cifrat cu DES în $C^{(i)}$, folosind înlănțuirea $II_0^{(i)} = OI_0^{(i-1)}$.

5.4.5. PROTOCOALE DE PROTECȚIE ȘI AUTENTIFICARE BAZATE PE CIFRURI GREU INVERSABILE

Un cifru greu inversabil (one-way) este reprezentat de o funcție neinversabilă, f, definită pe mulțimea mesajelor și cu valori în mulțimea criptogramelor. Este computațional imposibil să se determine în mod sistematic mesajul M din criptograma $C = f(M)$.

Cifrurile greu inversabile sînt utilizate în acele aplicații care nu necesită descifrarea datelor. O clasă de astfel de aplicații o constituie determinarea corespondenței între un mesaj M și o criptogramă C memorată în sistem. Corespondența se determină calculînd $f(M)$ și comparînd rezultatul cu C.

Un asemenea cifru poate fi implementat, de exemplu, prin utilizarea unui algoritm de cifrare bloc E:

$$f(M) = E_M(M_0),$$

unde M_0 este un mesaj standard (fix). Mesajul servește drept cheie pentru E. Atîta timp cît E este sigur, este computațional imposibil să se determine cheia M examinînd perechi $(M_0, E_M(M_0))$.

În [D-7] se sugerează implementarea cifrurilor greu inversabile utilizînd un polinom de forma:

$$f(x) = (x^n + a_{n-1}x^{n-1} + \ldots + a_1x + a_0) \bmod p,$$

unde p este un număr prim mare, iar n este, de asemenea mare. Deoarece un polinom de grad n are cel mult n rădăcini, pot exista cel mult n mesaje cu același text cifrat. Timpul de inversare a lui f (adică de găsire a rădăcinilor) este $O(n^2(\log p)^2)$; pentru $n = 2^{24}$ și $p = 2^{64}$ acest lucru cere 10^{16} operații.

Vom prezenta în cele ce urmează cîteva aplicații posibile ale cifrurilor greu inversabile.

5.4.5.1. **Protocoale de autentificare a utilizatorilor**

Fișierele de cuvinte de trecere (parole) pot fi protejate prin intermediul unor cifruri greu inversabile utilizînd o schemă publicată de Needham [N-1]. În loc să se memoreze cuvintele de trecere ale utilizatorilor în clar, ele vor fi transformate printr-un cifru f și memorate ca text cifrat într-un fișier care nu poate fi descifrat decît de către administratorul sistemului. Fiecare intrare în fișierul de parole este formată dintr-o pereche de forma $(ID, f(P))$, unde ID reprezintă identificatorul utilizatorului, iar P parola. Pentru a fi admis în sistem un utilizator trebuie să furnizeze ID și P. Sistemul calculează $f(P)$ și verifică dacă coincide cu valoarea înscrisă în fișierul de parole. Deoarece parola memorată nu poate fi descifrată, sistemul este sigur chiar dacă fișierul de parole este consultat (accidental sau voit).

În practică utilizatorii își aleg parole scurte și ușor de memorat. Astfel de parole sînt ușor de determinat prin încercări exhaustive. Se generează secvențe sau litere în mod sistematic, se cifrează și se compară rezultatul cu valorile din fișierul de parole. Într-un studiu privind securitatea parolelor la Laboratoarele UNIX, Morris și Thomson au ajuns la concluzia că aproximativ 86% din parole sînt ușor de compromis. În urma acestei constatări programul de introducere a parolei a fost modificat, astfel încît să încurajeze utilizarea unor parole lungi. Totodată, s-a convenit ca fiecare parolă să fie concatenată înainte de cifrare cu un număr aleatoriu de 12 biți. Cînd se creează o parolă P, un număr X este generat și concatenat la P. Dacă PX este rezultatul concatenării, se va memora în fișier atît X, cît și $f(PX)$, alături de identificatorul utilizatorului ID. Această îmbunătățire, deși nu face să crească factorul de muncă pentru găsirea parolei unui anumit utilizator (X fiind memorat neprotejat), are rolul de a crește factorul de muncă pentru generarea parolelor printr-o căutare exhaustivă: dacă parola are lungimea de n biți, există 2^{n+12} posibilități pentru $f(PX)$, adică de 2^{12} ori mai multe decît în situația cînd se cifrează doar parola. Însă introducerea valorilor P și ID în clar de la terminale sau calculatoare și transmiterea lor prin rețea poate duce la compromiterea lor. De aceea Feistel, Notz și Smith [F—1] au publicat o pro-

cedură de autentificare care nu duce la expunerea parolei utilizatorului și permite autentificarea mutuală atît a utilizatorului, cît și a sistemului. Vom presupune că fiecare utilizator A are o cheie privată (de exemplu, pe cartelă magnetică) ce poate fi inserată pe terminalul la care lucrează A. De asemenea această cheia este memorată și într-un fișier al sistemului. Protocolul de autentificare cu parole (ilustrat și în fig. 5.20) este următorul:

— utilizatorul A transmite sistemului S identificatorul $ID = A$;

— S transmite la A: $X = E_A(T)$, unde T este data curentă, iar E_A — o transformare criptografică derivată din cheia privată a lui A;

— A va descifra X și va obține T, pe care îl va compara cu data zilei. Dacă verificarea corespunde, A va emite la S:

$$Y = E_A(T, P),$$

unde P este parola lui A;

— S va descifra Y pentru a obține T și $P \cdot T$ va fi comparat cu valoarea emisă în pasul (2), iar $f(P)$ se va compara cu valoarea din fișierul de parole.

Protocolul poate fi modificat și pentru un sistem cu chei publice. În pasul (2) sistemul S va folosi transformarea sa secretă D_S (de autentificare) pentru a calcula $X = D_S(T)$, semnătura pe care A o va valida utilizînd transformarea publică a lui S, E_S. În pasul (3) A folosește transformarea publică a lui S (pentru secretizare), determinînd $Y = E_S(T, P)$. Numai S va putea descifra criptograma, obținînd parola P a lui A.

Este posibil să fie imaginat un protocol de autentificare bazat pe semnătura digitală, fără să se utilizeze parole:

— A va transmite lui S: $ID = A$;

— S va transmite la A:

$$X = D_S(T),$$

unde T este data curentă, iar D_S transformarea secretă a lui S;

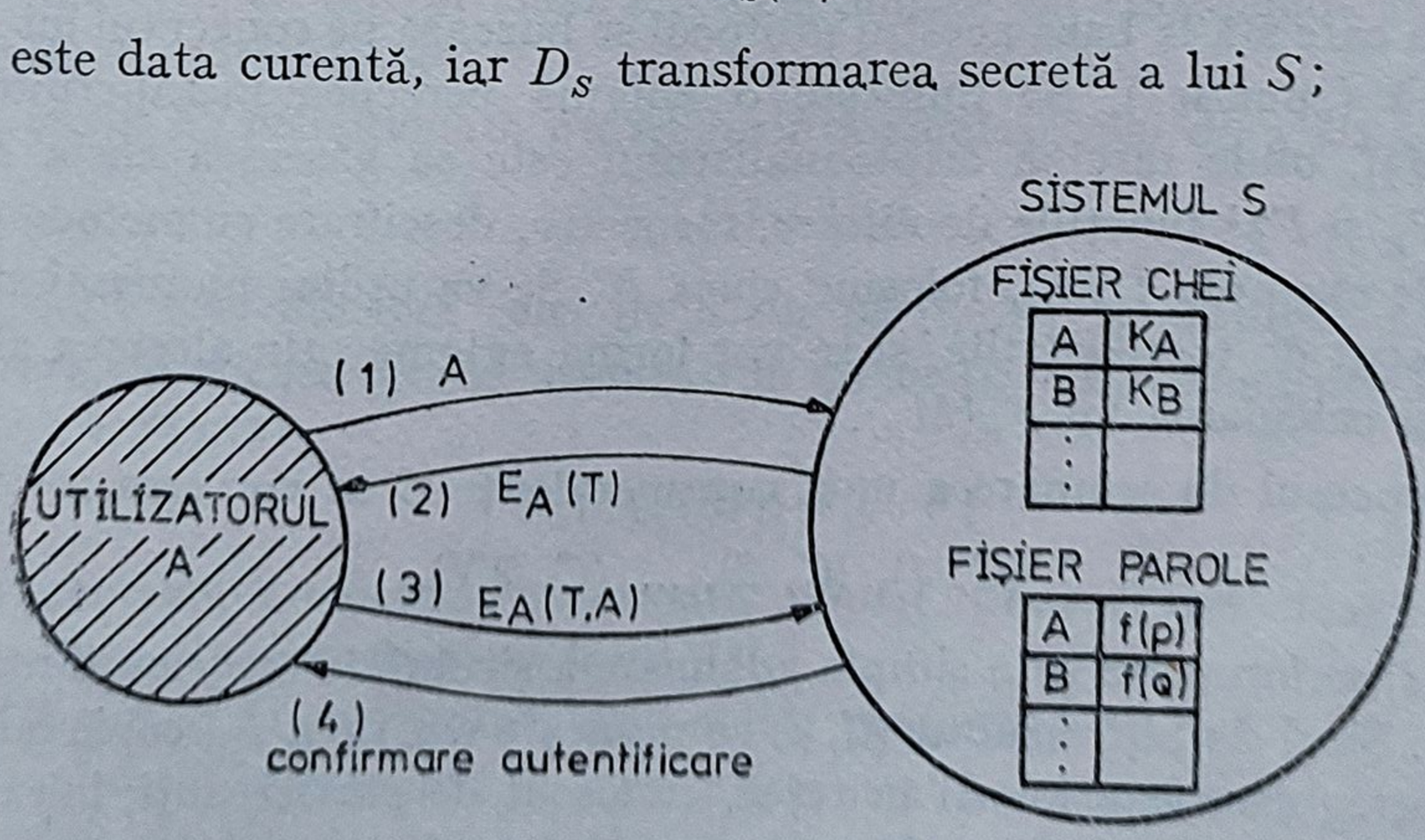


Fig. 5.20. Protocol de autentificare cu parole

— A calculează $E_S(X) = T$, utilizînd cheia publică a lui S și verifică T. Dacă corespunde, A va emite la S:

$$Y = D_A(T),$$

în care D_A este transformarea secretă a lui A;

— sistemul validează Y, folosind transformarea publică E_A a lui A.

Cele două protocoale de autentificare, cu parole și cu semnătură digitală, pot fi combinate într-un mecanism hibrid de autentificare:

— A emite la S: $ID = A$;

— S emite la A: $X = D_S(T)$;

— A, după verificarea lui T, va emite la S:

$$Y = D_A(T),\ E_S(T, P);$$

— Sistemul S validează accesul utilizatorului doar dacă corespunde atît semnătura $D_A(T)$, cît și parola înscrisă în fișierul cu parole.

5.4.5.2. Protocol pentru semnătură digitală bazat pe chei publice

Acest protocol are la bază ideea de randomizare și utilizează în vederea cifrării atît algoritmi cu chei publice, cît și convenționali [D—5].

Fie E_P și D_P algoritmi de cifrare, respectiv, de descifrare cu cheie publică ai unui participant P. Cum este cunoscut, E_P este făcut public, pe cînd D_P este cunoscut doar de către P.

Pentru fiecare mesaj M se definește semnătura lui P ca fiind:

$$S(M) = D_P(M).$$

Partea care verifică semnătura folosește E_P făcut public, determinînd:

$$E_P(D_P(M)) = M.$$

Ideea care stă la baza acestui protocol se bazează pe conceptul lui Merkle de *enigmă* („puzzle"). Se numește o enigmă perechea mesaj-criptogramă, $(M, F_K(M))$, unde sarcina criptoanalistului este să găsească cheia K. Vom nota cu F_K și F_K^{-1} funcțiile de cifrare, respectiv, descifrare cu metode convenționale (de exemplu DES), folosind cheia K. Se va utiliza un mesaj standard M, cunoscut de toate părțile, și se vor forma enigme prin alegerea aleatoare a lui K și calcularea lui $F_K(M_S)$.

În procesul de semnare a unei corespondențe C se vor crea $2N$ mesaje:

$$(C, 1),\ (C, 2),\ \ldots,\ (C, 2N),$$

unde (C, i) se formează prin simpla adăugare a șirului de biți care îl reprezintă pe i la C. Șirul de biți rezultat (C, i) se numește cea de a i-a copie a lui $C.N$ este un întreg care este folosit în rețea, cunoscut de participanți; în exemplele noastre se va considera $N = 50$.

Să convenim că A are obligația formală din partea lui B de a accepta C dacă el posedă orice $N + 1$ din cele $2N$ semnături ale lui B:

$$D_B(C, 1),\ D_B(C, 2),\ ...,\ D_B(C, 2N).$$

Protocolul funcționează după următorii pași:

(1) **Procedura semnăturii**

(i) Utilizatorul (procesul) A pregătește $2N$ enigme:

$$F_{K_1}(M_S),\ F_{K_2}(M_S),\ ...,\ F_{K_{2N}}(M_S)$$

și $2N$ copii semnate și cifrate ale lui C:

$$F_{K_1}(D_A(C, 1)),\ F_{K_2}(D_A(C, 2),\ ...,\ F_{K_{2N}}(D_A(C, 2N))$$

și le emite pe toate la B;

(ii) Utilizatorul (procesul) B alege un $1 \leqslant i \leqslant 2N$, nefolosit anterior. El îi va cere lui A să îi releve cheia sa K_i. A va trimite cheia cerută. După recepția lui K_i, B poate verifica dacă cheia este într-adevăr soluția celei de a i-a enigme a lui A. El utilizează apoi K_i pentru descifrarea semnăturii lui A de pe cea de a i-a copie și verifică că este într-adevăr semnătura lui A asupra lui (C, i), prin aplicarea funcției E_A la $D_A(C, i)$. Pasul (ii) se repetă de $N + 1$ ori. B acționează simetric.

Fiecare pereche $F_{K_i}(M_S)$, $F_{K_i}(D_A(C, i))$ este numită cea de i-a unitate a mesajului emis de A către B în pasul (i).

(2) **Autentificarea semnăturii**

După recepția lui K_i de la A, ca urmare a cererii lui B, acesta consideră unitatea ca fiind falsă dacă are loc una din situațiile:

— numărul de biți ai cheii K_i recepționate nu corespunde cu cel fixat la stabilirea protocolului;

— cheia K_i recepționată nu reprezintă soluția celei de a i-a enigme P_i:

$$F_{K_i}^{-1}(P_i) \neq M_S;$$

— după decodificarea celei de a i-a părți, M_i, a pretinsei secvențe de semnătură cifrată pe care A a emis-o în pasul (i) la B, semnătura lui, (C, i'), nu este verificată:

$$E_A(F_{K_i}^{-1}(M)) \neq (C, i')$$

unde i' reprezintă valoarea anterioară a lui i. Cînd B detectează o unitate falsă el întrerupe imediat protocolul.

În figura 5.21 se prezintă sistematic protocolul propus.

O aplicație interesantă a acestui protocol este în domeniul poștei electronice din rețelele teleinformatice și vizează semnarea unui contract C de către doi utilizatori A și B. În acest caz corespondența C reprezintă contractul

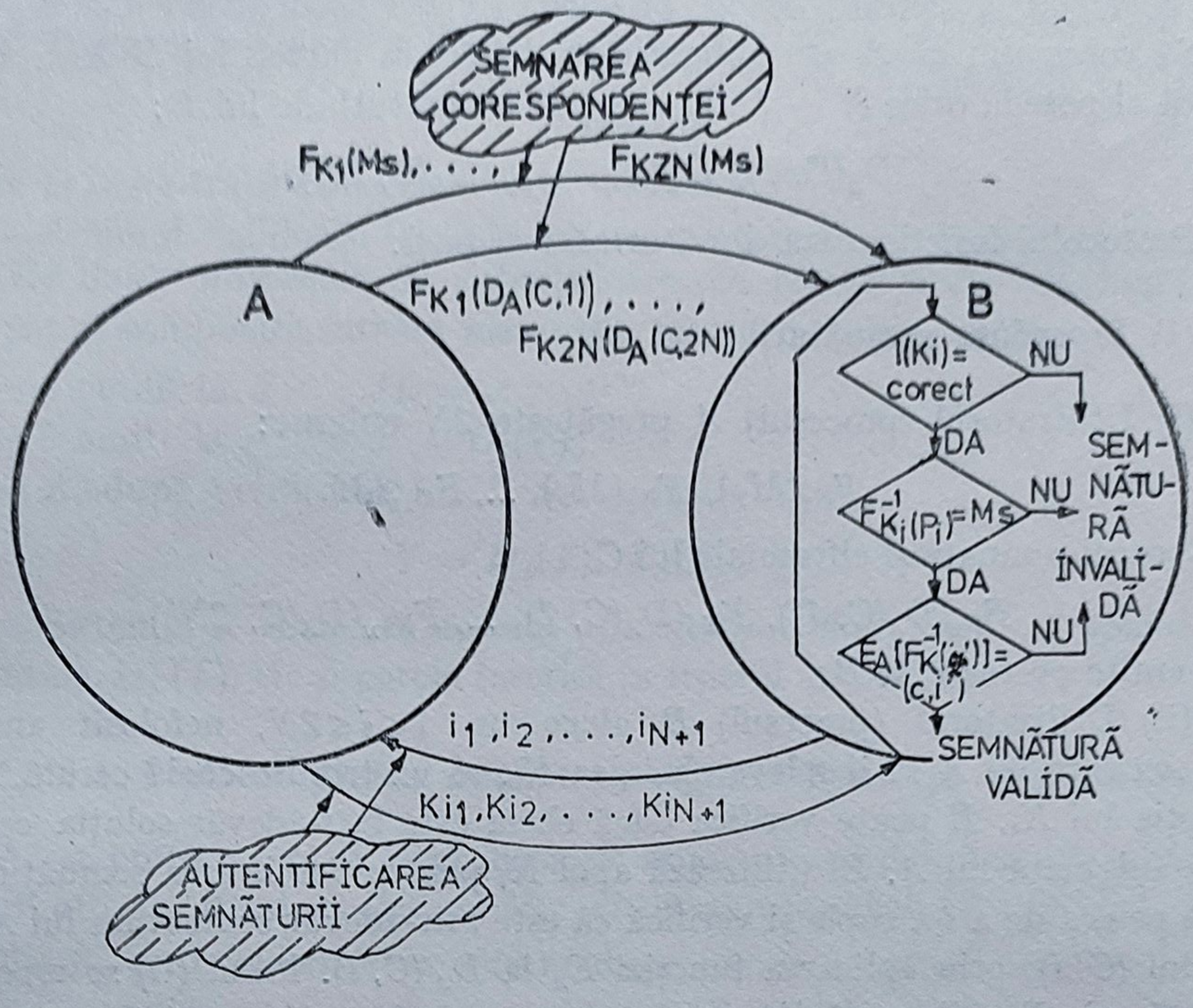


Fig. 5.21. Protocol de semnătură digitală bazat pe conceptul de cheie publică

ce va fi emis și semnat atît de la *A* către *B*, cît și în sens invers. Se poate observa că protocolul evită situația cînd una din părți întrerupe protocolul în momentul în care a obținut semnătura celeilalte părți fără a-i trimite și propria-i semnătură.

5.4.6. INTEGRAREA SISTEMELOR CRIPTOGRAFICE ÎN ARHITECTURA ISO

Datele care sînt protejate prin cifrare este posibil să fie transmise de-a lungul mai multor legături în cadrul rețelei pînă să ajungă la destinație. De exemplu, datele pot constitui intrări de la un terminal către un program utilizator ce le prelucrează și le memorează într-un fișier pe disc. Mai tîrziu, utilizatorul va regăsi datele și le va afișa pe terminal. În rețelele de calculatoare datele trebuie transmise de la anumite entități la altele în vederea prelucrării și memorării.

Există mai multe posibilități de integrare a punctelor de cifrare în arhitectura unei rețele. O variantă este așa-numita cifrare la nivel legătură de date, cînd mesajul M este cifrat și descifrat pe fiecare nod între nodul sursă și nodul destinație (fig. 5.22). Mesajul este prelucrat în clar de nodul i și transmis cifrat, sub forma $E_i(M)$, pe cea de a i — legătură, către nodul următor $(i + 1)$. Fiecare legătură i are propria pereche de transformări, E_i și D_i, care pot diferi și ca algoritmi și ca format de mesaj. În acest caz este protejată doar transmisia pe suportul fizic al rețelei.

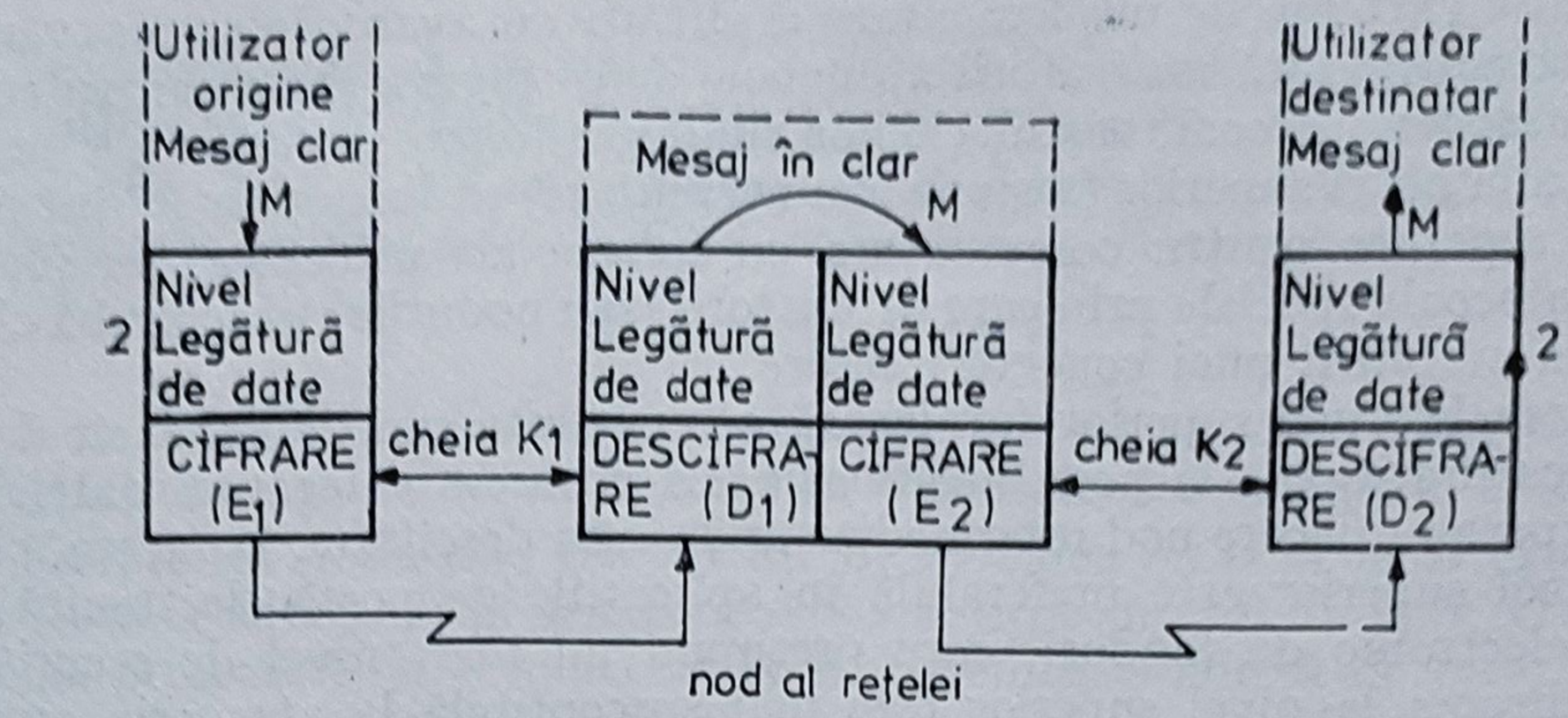


Fig. 5.22. Cifrarea la nivel legătură de date

Cifrarea la nivel rețea reprezintă o versiune modificată a celei la nivel legătură de date, restrîngîndu-se circulația mesajului în clar într-un „modul de securitate" (fig. 5.23). Fiecare legătură utilizează o cheie unică, iar trecerea de la o cheie la alta se face în modulul de securitate.

Cifrarea la nivel superior este o soluție ce pare mai naturală într-un sistem orientat pe „sesiuni", unde schimburile sînt relativ lungi. Toate informațiile care tranzitează prin rețea sînt cifrate, în general, cu aceeași cheie pe durata unei sesiuni între entitățile sursă și destinație ale schimbului. Cifrarea poate fi implementată prin protocoale la nivel sesiune sau prezentare (fig. 5.24).

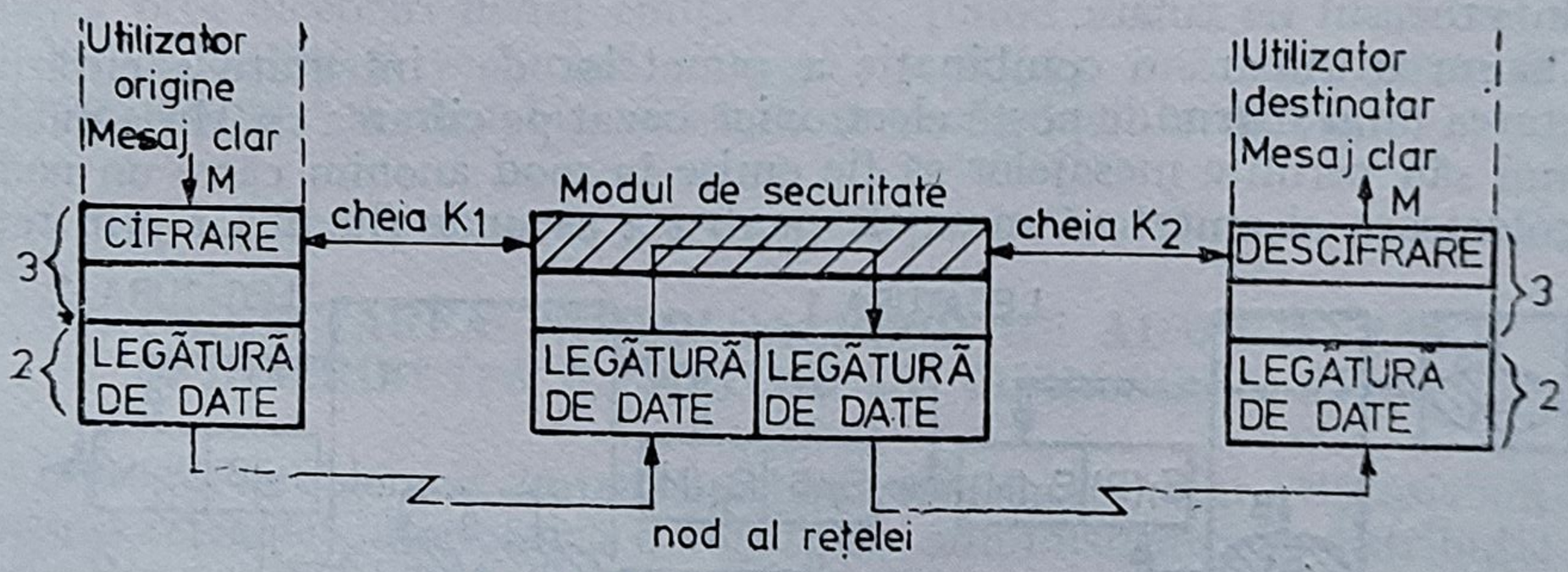


Fig. 5.23. Cifrarea la nivel rețea

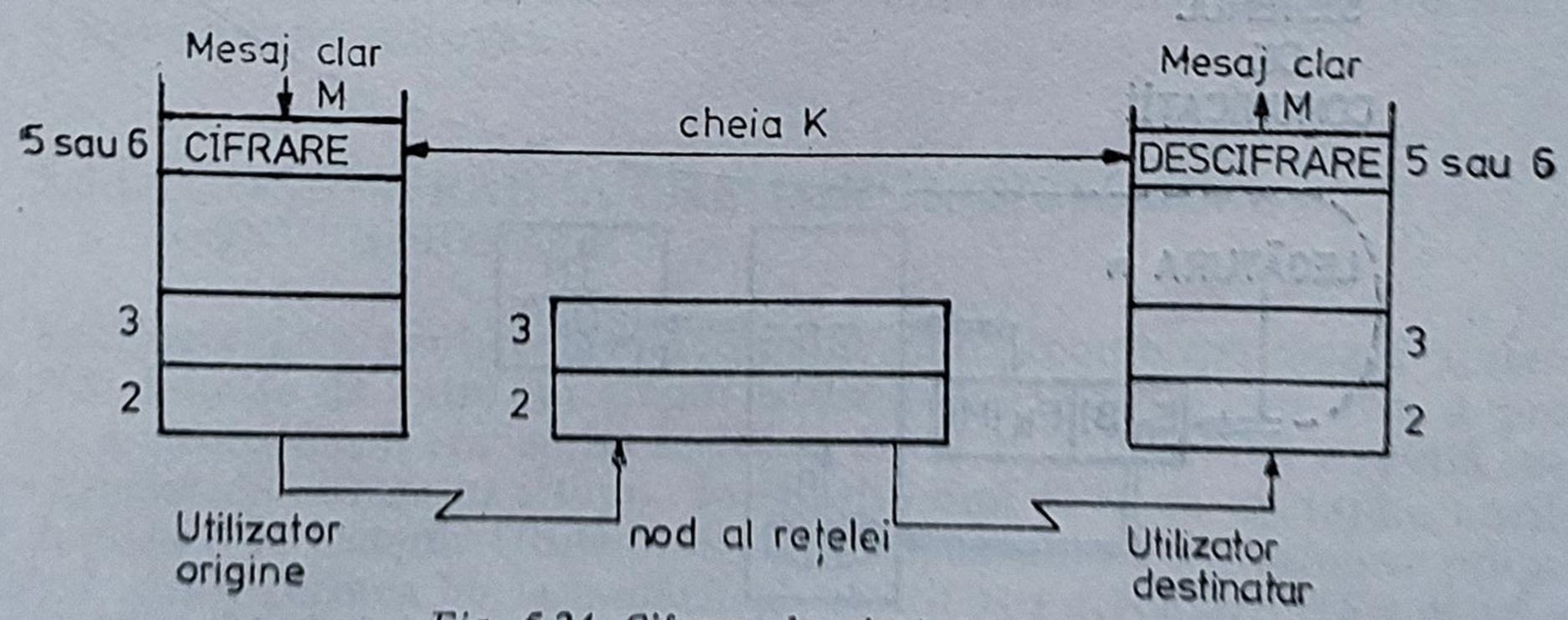


Fig. 5.24. Cifrarea la nivel superior

Soluțiile extreme de implementare a cifrării au avantaje și dezavantaje La cifrarea pe legătură utilizatorii au nevoie de o singură cheie pentru comunicarea cu sistemul local, restul cheilor fiind gestionate automat de sistem. La cifrarea la nivel superior (sesiune sau prezentare) utilizatorilor le sînt necesare chei separate pentru comunicarea cu fiecare alt utilizator. În plus se impun protocoale speciale prin care utilizatorii (sau nodurile) își schimbă cheile în vederea stabilirii unei conectări sigure.

Cifrarea la nivel superior furnizează o securitate sporită, întrucît datele nu sînt descifrate pînă la nodul destinație. La cifrarea pe legătură datele puteau fi expuse în fiecare nod intermediar unde erau descifrate. De aceea, cifrarea la nivel superior este preferabilă în aplicațiile de poștă electronică sau transfer electronic de fonduri, care presupun un înalt nivel de securitate. Totuși, cifrarea la nivel superior este mai susceptibilă la atac prin analiza traficului de mesaje. Aceasta deoarece — spre deosebire de cifrarea pe legături, unde adresele nodurilor destinație erau transmise cifrat de-a lungul legăturii — adresa destinatarului trebuie palsată în clar, fiind necesară procesului de rutare a mesajelor.

În cazul unor aplicații care solicită un înalt nivel de securitate cele două moduri extreme de abordare prezentate pot fi combinate. În figura 5.25. se arată cum este posibil să fie utilizat un sistem de cifrare la nivel superior în combinație cu cifrarea pe legătură a adresei de destinație. Fiecare pachet emis pe legătură are două cîmpuri: un cîmp antet și unul de date. Antetul conține adresa de destinație a mesajului, cifrată cu cheia de legătură cu nodul vecin, plus alte informații de control. În cadrul fiecărui nod intermediar mesajul trece cifrat, singura informație descifrată fiind adresa de destinație necesară în procesul de rulare.

Chaum utilizează o combinație a punctelor de integrare a cifrării în proiectarea unui sistem de poștă electronică bazat pe cifrarea cu cheie-publică. Sistemul său permite mesajelor să fie emise în mod anonim către un nod *S*, care colectează și amestecă mesajele pentru a ascunde fluxul lor prin rețea.

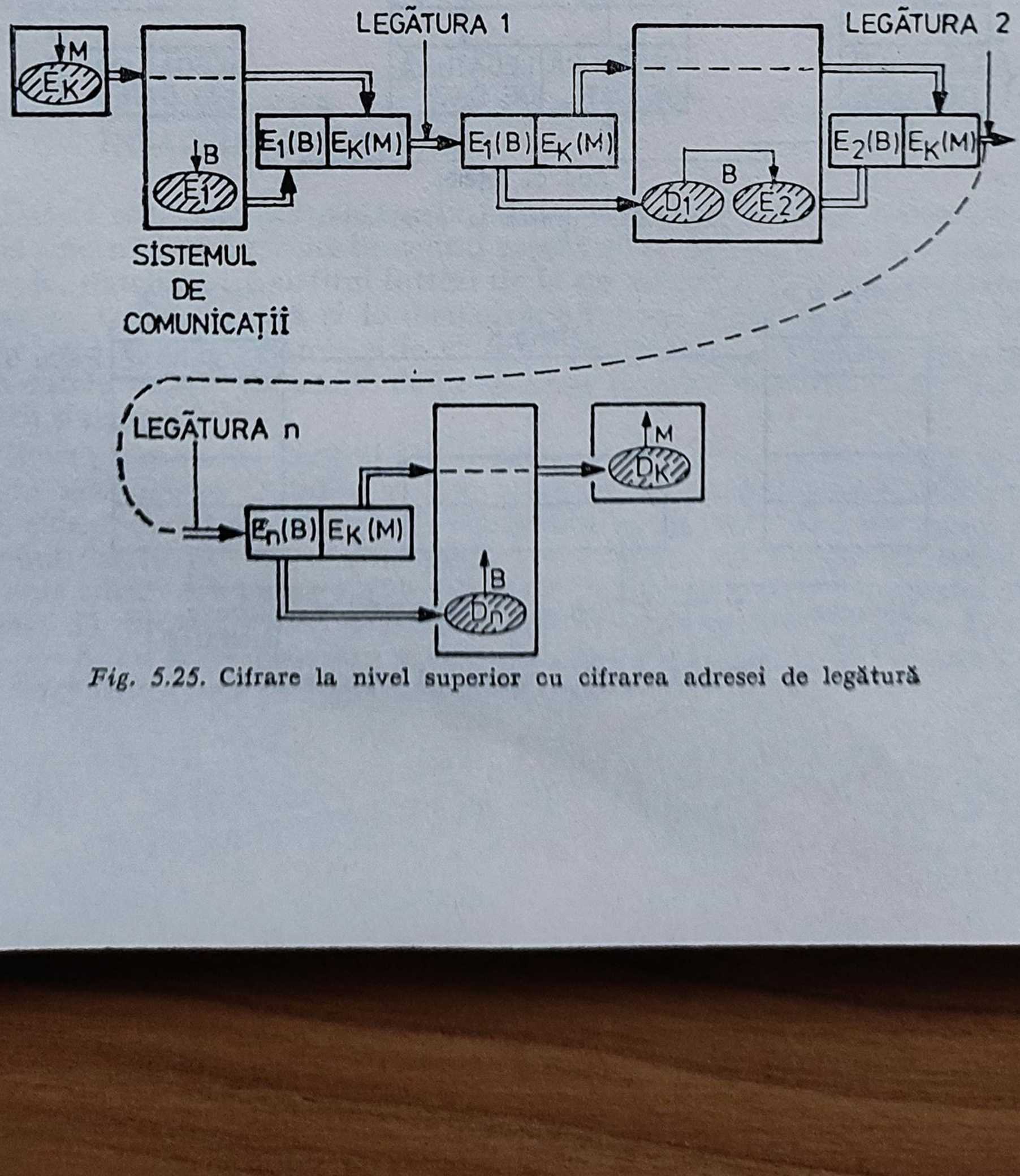


Fig. 5.25. Cifrare la nivel superior cu cifrarea adresei de legătură

Să presupunem că utilizatorul A dorește să transmită un mesaj anonim M, la utilizatorul B. Mai întîi A va cifra pe M folosind transformarea publică a lui B, E_B. Apoi A va cifra destinația B și mesajul cifrat $E_B(M)$ utilizînd transformarea publică a lui S, E_S, și transmite la S:

$$C = E_S(B, E_B(M)).$$

S va descifra C folosind transformarea secretă D_S și va direcționa mesajul $E_B(M)$ către B, care-l va descifra cu transformarea sa secretă D_B.

Emițătorul A nu este relevat lui B, iar calea de la A la B este ascunsă prin cifrarea adresei B pe legătura $A—S$ și prin antetul de la S. Dacă A dorește să primească un răspuns de la B, va emite o adresă de retur, neinteligibilă pentru B, împreună cu o cheie K, pe care B o va folosi să emită răspunsul. Mesajul transmis de A lui S va cuprinde:

$$C = E_S(B, E_B(M, U, K)), \quad \text{unde}$$

$U = E_S(A)$ reprezintă adresa de retur cifrată cu cheia nodului S. B va putea să emită rsăpunsul M' lui S:

$C' = E_S(U, E_K(M'))$. Apoi S va descifra pe U și va emite mai departe la A pe $E_K(M')$.

În această schemă A nu poate semna mesajul M, pentru a nu-i divulga identitatea, ceea ce impune să se pună întrebarea dacă altcineva nu este capabil să determine pe M, încercînd diverși X, pînă cînd $E_B(X) = E_B(M)$ (transformarea E_B este făcută publică). De asemenea, B poate fi capabil să găsească prin încercări nodul emițător, A, ținînd seama de numărul relativ mic de posibilități și verificînd că $E_S(Y) = E_S(A)$. Problema poate fi rezolvată prin adăugarea unor biți aleatorii la mesaj înainte de cifrare.

5.5. PROIECTAREA PROTOCOALELOR DE ADMINISTRARE ȘI DISTRIBUȚIE A CHEILOR DE CIFRARE ÎN STI

Una din problemele importante în proiectarea unui criptosistem sigur pentru o rețea teleinformatică o constituie administrarea și distribuția cheilor de cifrare. Chiar dacă algoritmul de cifrare este computațional imposibil de „spart", întregul sistem poate deveni vulnerabil dacă cheile nu sînt protejate într-un mod adecvat.

5.5.1. ADMINISTRAREA CHEILOR ÎN SISTEMELE CONVENȚIONALE

Vom considera pentru început modul cum trebuie gestionate acele chei care sînt folosite de către un singur utilizator (sau proces) pentru a proteja datele ce urmează a fi transmise în rețea. O primă cale este să se evită memorarea cheilor de cifrare în sistem. În subsistemul IPS, de exemplu, cheile nu sînt rezidente în sistem. Utilizatorii sînt responsabili cu gestiunea propriilor chei și cu introducerea lor la momentul cifrării sau descifrării. IPS oferă utilizatorilor două formate de introducere a cheilor de 56 de biți cerute pentru

DES. Primul format constă din introducerea directă a celor 8 octeți ai cheii (56 de biți, plus 8 biți de paritate). Se recomandă evitarea unor chei formate din cuvinte sau asocieri de litere și cuvinte, ce sînt ușor de găsit prin căutare exhaustivă. Deoarece este însă mai ușor pentru utilizatori să rețină șiruri cu înțeles, IPS furnizează și un al doilea format, în care cheile sînt introduse sub forma unor șiruri lungi de caractere. Șirul este redus la 56 biți prin cifrare cu DES utilizînd înlănțuirea blocurilor și reținînd ultimii 56 de biți.

Cheile pot fi reținute și într-o memorie ROM sau pe o cartelă magnetică, care este posibil să fie citită printr-un dispozitiv special.

În sistemele de cifrare convențională utilizatorii trebuie să-și înregistreze cheia secretă la calculator, în scopul stabilirii unui canal sigur între terminalele lor și calculatorul central. Aceste chei, memorate în sistem, vor trebui protejate; cea mai simplă metodă este să se memoreze într-un fișier cifrat cu o *cheie master*.

În sistemele bazate pe chei publice un utilizator A nu trebuie să-și înregistreze transformarea sa secretă D_A la calculatorul cu care urmează să stabilească un canal de comunicații sigur. Chiar dacă D_A are protecția asigurată este nevoie de un mecanism care să rezolve situațiile cînd este compromisă (dacă se folosește semnătură digitală). În acest scop există un administrator de semnături S care reține într-un articol atît vechea, cît și noua transformare publică E_A precum și perioada cînd aceste semnături sînt valide. Pentru a determina dacă un mesaj a fost semnat înaintea compromiterii mesajele trebuie să poarte data emiterii. Administratorul S este cel care va atașa data mesajelor semnate de A, sub propria-i semnătură D_S, jucînd astfel rolul unui *notar public*. Un mesaj M va fi deci dublu semnat:

$$C = D_S(D_A(M), T).$$

Un alt utilizator, recepționînd C, va putea verifica, prin intermediul lui S dacă semnătura lui A este validă momentului T înregistrat în C (dacă nu s-a folosit pentru semnătura transformarea D_A deja compromisă la acel moment). O strategie similară poate fi folosită la protecția semnăturii D_A într-un sistem convențional. Deoarece însă transformările corespunzătoare de cifrare E_A și E_S sînt secrete, receptorul unui mesaj semnat $C = D_S(D_A(M))$ nu poate valida semnătura.

Ehrsam, Matyas, Meyer și Tuchman [E—1) descriu un sistem complet de administrare și distribuție a cheilor pentru protecția comunicațiilor și a fișierelor în rețeaua SNA. Dacă datele sînt protejate prin cifrare (prin algoritm DES), securitatea datelor este funcție de secretul cheilor. Cu alte cuvinte cheile nu trebuie să apară în clar în afara dispozitivelor de cifrare (modul de securitate), cu excepția unui scurt moment corespunzător generării și instalării lor. Dacă o cheie este memorată sau distribuită către un alt nod trebuie protejată prin cifrare cu o altă cheie.

Cheile sînt de două categorii: de cifrare a datelor și de cifrare a cheilor. La rîndul lor, cheile de cifrare a cheilor se împart în *primare* (master) — *KM* —, care protejează cheile de date sau alte chei de cifrare a cheilor (sînt memorate în clar), și *secundare*, ce sînt utilizate doar la cifrarea cheilor de date și sînt memorate cifrate sub o cheie primară.

În cadrul lui SNA se definește un *punct de control sistem* (SSCP), care asigură funcția de control tuturor activităților de comunicații pentru o porțiune a rețelei, numită *domeniu*. În timpul procesului de definire tabelele sînt construite astfel încît să descrie pentru SSCP configurația și caracteristicile

domeniului său, inclusiv caracteristicile de cifrare ale resurselor din domeniu. O funcție de gestiune a cheilor este localizată în SSCP spre a produce o tabelă de chei pentru domeniul respectiv. În acest fel SSCP își poate asuma rolul de gestiune a resurselor de cifrare din domeniul său.

Comunicațiile dintre domenii (între SSCP-uri) necesită schimbarea unor chei între SSCP-uri. În acest scop, o cheie secundară de cifrare a cheilor KNC este utilizată la cifrarea cheilor de date. În SNA ea este numită *cheie de domeniu*. În mod uzual asemenea chei sînt definite perechi, una pentru transmiterea cheilor de date, iar alta pentru recepția lor.

Vom presupune că fiecare calculator gazdă (host) are o *cheie master*, *KMO*, cu două variante, *KM1* și *KM2*. Variantele pot fi simple funcții ale lui *KMO*. Cheia master este folosită la cifrarea altor chei de cifrare și la generarea unor noi chei de cifrare. Fiecare calculator terminal are o cheie master a terminalului *KMT*, care furnizează un canal sigur între terminal și calculatorul gazdă în vederea schimbului de chei. Sistemul memorează copii ale acestor chei (*KMT*) într-un fișier cifrat sub cheia *KM1*. Alte chei de cifrare a datelor, cum ar fi cheile master de fișier (numite chei secundare de fișier) sînt memorate în fișiere cifrate cu *KM2*.

Cheia master *KMO* este memorată, pe o memorie nevolatilă, într-un dispozitiv hardware special numit modul de securitate; KM1 și KM2 sînt calculate pornind de la KMO. Cheile de cifrare a datelor sînt memorate și transmise modulului de securitate sub formă cifrată. Modulul de securitate furnizează o serie de operații speciale pe calculatorul gazdă sau pe terminal.

Pe calculatorul gazdă se pot utiliza de către sistemul de gestiune a cheilor următoarele operații:

• *Instalarea cheii master* (*smk*). Cheia *KMO* este instalată cu operația *smk* (*KMO*). Această operație cere o protecție specială.

• *Cifrare sub cheia master* (*emk*). O cheie K este protejată prin cifrare cu cheia *KMO* cu ajutorul operației *emk* $(K) = E_{KMO}(K)$.

• *Cifrare* (*ecph*). Pentru a cifra un mesaj M utilizînd cheia K, aceasta este transmisă modulului de securitate cifrată cu *KMO*. Dacă $X = E_{KMO}(K)$ este cheia cifrată, mesajul M va fi cifrat cu operația: *ecph* $(X, M) = E_K(M)$, unde $K = D_{KMO}(X)$ (fig. 5.26). Cheia K este transmisă modulului de securitate sub formă cifrată.

• *Descifrare* (*dcph*). În mod similar criptograma C este descifrată cu cheia K cu ajutorul operației: *dcph* $(X, C) = D_K(C)$, unde $K = D_{KMO}(X)$ (fig. 5.27).

• *Recifrare de sub cheia master* (*rfmk*). O cheie master de terminal *KMT* este memorată cifrat cu *KM1*, $W = E_{KM1}(KMT)$. Recifrarea de sub cheia

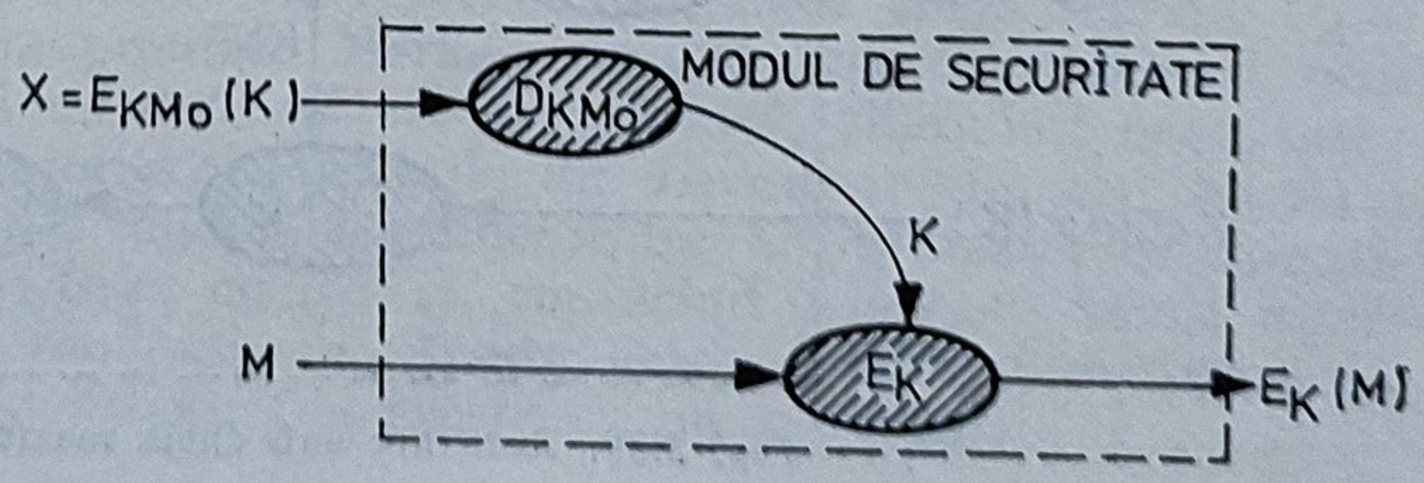


Fig. 5.26. Funcția de cifrare

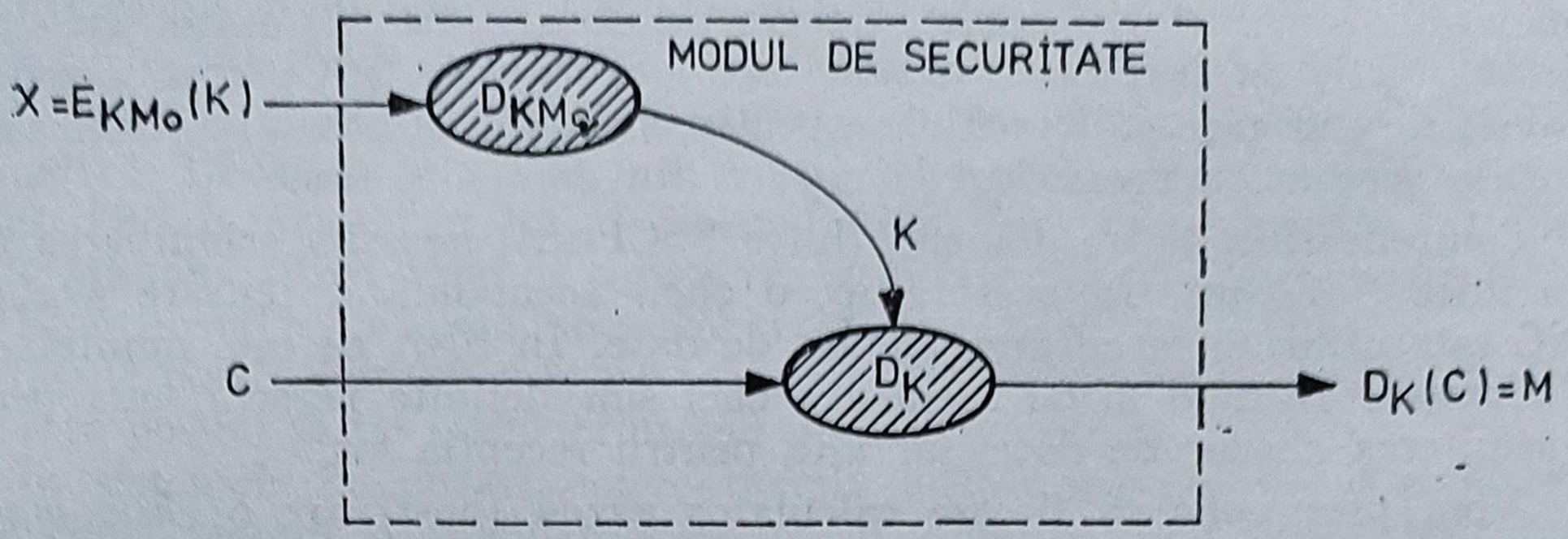


Fig. 5.27. Funcția de descifrare

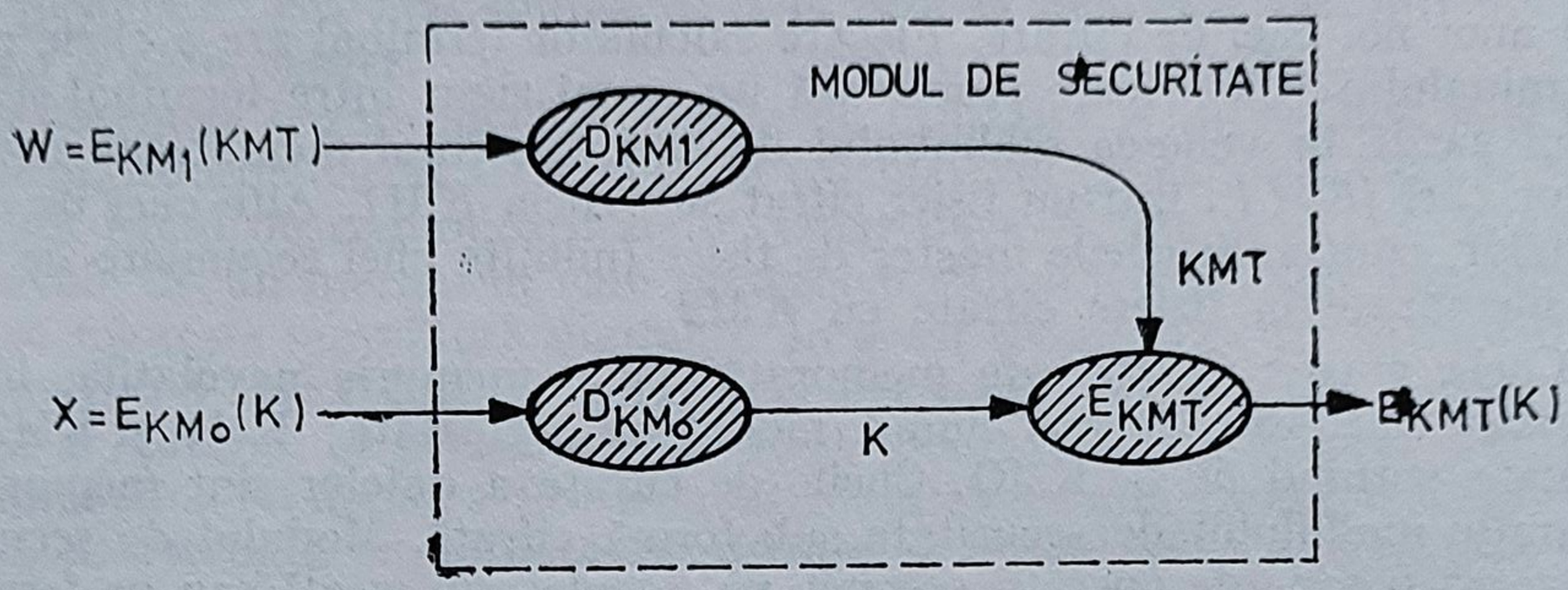


Fig. 5.28. Recifrare de sub cheia master

master permite să se treacă o cheie K — cifrată cu KMO, $X = E_{KMO}(K)$ — sub cheia KMT: $rfmk\ (W, X) = E_{KMT}(K)$, unde $KMT = D_{KM1}(W)$ și $K = D_{KMO}(X)$ (fig. 5.28). Operația este necesară cînd se transmit chei către terminale sau alte calculatoare gazdă în vederea unor transmisii cifrate de mesaje, fișiere etc. KMT se memorează sub cheia $KM1$ și nu sub KMO, $W' = E_{KMO}(KMT)$, deoarece K poate fi obținută din $Y = E_{KMT}(K)$ prin funcția de descifrare $dcph(W', Y) = D_{KMT}(Y) = K$.

• *Recifrare sub cheia master* (*rtmk*) .O cheie K de protecție a unui fișier nu se cifrează cu KMO, deoarece este posibil să fie determinată cu funcția *dcph*. Ca urmare, K se memorează prin cifrare cu o cheie secundară de fișier KNF, $X = E_{KNF}(K)$. Cheia KNF se memorează sub $KM2$, $W = E_{KM2}(KNF)$. Pentru a se utiliza K trebuie cifrată sub cheia master (KMO) prin funcția $rtmk\ (W, X) = E_{KMO}(K)$, unde $K = D_{KNF}(X)$ și $KNF = D_{KM2}(W)$.

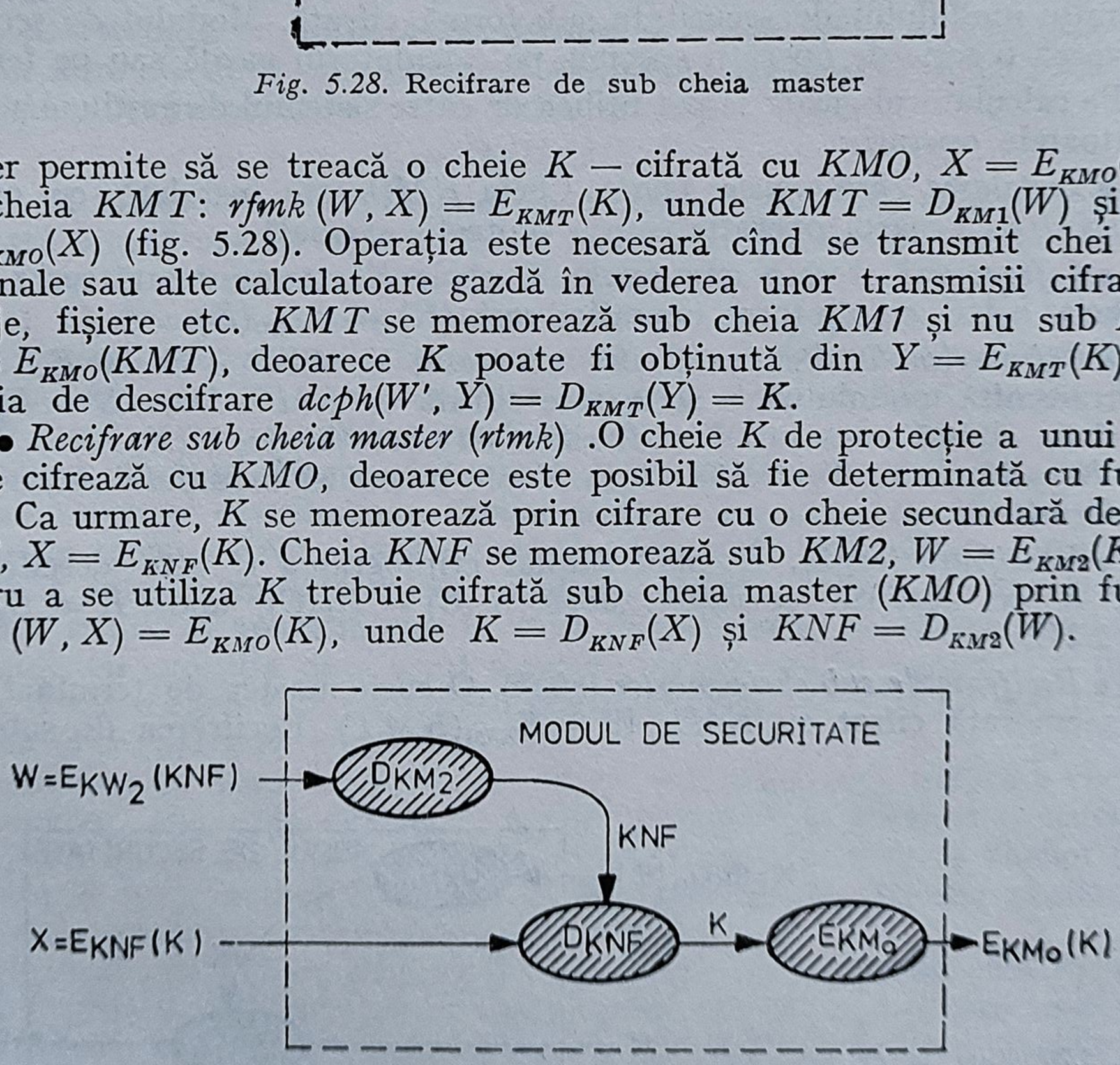


Fig. 5.29. Recifrare sub cheia master

La *terminal* cheia master KMT este memorată în modulul de securitate, care execută următoarele operații:

• *Descifrare de sub cheia master* (*dmk*). Această operație obține cheia K transmisă la terminal cifrată, $E_{KMT}(K)$, și o memorează într-un registru special al modulului de securitate, unde rămîne pe tot timpul transmisiei de date pînă la o eventuală schimbare a ei.

• *Cifrare* (*ecph*). Operația permite cifrarea datelor folosind cheia K memorată în registru. Datele cifrate sînt transmise în rețea.

• *Descifrare* (*dcph*). Operația facilitează descifrarea datelor cu ajutorul cheii K memorate în registru.

Comunicarea între entități ale rețelei (programe de aplicații și terminale) este bazată pe o legătură logică numită sesiune. După ce s-a stabilit o sesiune între două entități ale rețelei, informația poate fi transmisă într-un sens sau altul. Criptografia este specificată prin parametri în timpul fazei de inițializare.

5.5.2. ADMINISTRAREA CHEILOR PUBLICE

Cheile publice pot fi distribuite la utilizatori sub formă de copii, după ce au fost memorate pe un mediu extern, de exemplu cartelă magnetică. Altă modalitate de distribuție este aceea efectuată prin intermediul unui fișier unde sînt memorate și gestionate de către un administrator, S. Protecția cheilor nu este necesară întrucît ele nu sînt secrete. În schimb menținerea integrității lor este obligatorie pentru ca autentificarea făcută cu ele să fie validă. Un utilizator rău intenționat are posibilitatea să schimbe una din cheile publice și — ca atare — să descifreze o anumită clasă de mesaje cifrate. Situația este prevenită dacă se cere o semnătură de la S. Konfelder propune o soluție bazată pe *certificate*. După înregistrarea transformării publice E_A la sistem, utilizatorul A va primi un certificat semnat de S, care-l conține pe S_A. Certificatul lui A are forma·

$$C_A = D_S(A, E_A, T),$$

unde T reprezintă data curentă (eventual și ora), iar D_S — transformarea secretă a lui S. A poate verifica dacă certificatul provine de la S și conține cheia publică corectă calculînd:

$$E_S(C_A) = (A, E_A, T).$$

prin utilizarea transformării publice E_S a lui S. De asemenea, prin dată și oră se verifică dacă certificatul este curent.

Una din problemele dificile de rezolvat este dacă cheia secretă D_S a administratorului este compromisă. Merkele rezolvă această problemă prin introducerea conceptului de *autentificare arborescentă*, eliminînd necesitatea semnăturii din partea autorității. Intrările din fișierul public sînt semnate ca un tot unitar, utilizînd o funcție de randomizare. Un utilizator își autentifică propria cheie într-un fișier cu n chei, cunoscînd 0 ($\log_2 n$) valori intermediare ale funcției de randomizare. Aceste valori intermediare formează o cale de autentificare în arbore și servește ca un certificat.

Fie K_1, K_2, ..., K_n intrările din fișierul de chei publice și $f(x, y)$ o funcție care returnează o valoare în domeniul lui f. Funcția de randomizare se definește succesiv astfel:

$$H(i, j) = \begin{cases} f(H(i, m), H(m+1, j)), \text{ dacă } i < j, \text{ unde } m = \left[\frac{i+j}{2}\right]; \\ f(K_i, K_i) \qquad\qquad \text{dacă } i = j, \end{cases}$$

unde $H(1, n)$ este valoarea corespunzătoare întregului fișier public.

5.5.3. GENERAREA CHEILOR DE CIFRARE BLOC

În paragraful 5.4.2. au fost analizate metodele de generare a unor chei pseudoaleatoare pentru cifrurile secvențiale. Șirul cheie era generat pornind de la o valoare inițială I_0, utilizînd un algoritm de cifru-bloc. Vom discuta în cele ce urmează modul de generare a unor chei individuale a căror valoare trebuie să satisfacă aceleași proprietăți de aleatorism ca și cheile-șir.

Matyas și Meyer [M-2] sugerează o metodă de generare a cheilor pentru cheile master *KMO*, *KM1* și *KM2*, descrise la protecția informațiilor în SNA. Ei recomandă ca *KMO* să fie generată în afara sistemului, printr-un proces aleatoriu. Cheia poate fi memorată pe un mediu magnetic și apoi instalată cu operația *smk*. O mulțime de chei de cifrare a cheilor, K_i $(i = 1, 2, ...)$, este derivată dintr-un număr aleatoriu, R, generat în afara sistemului în aceeași manieră ca și *KMO*. Fiecare K_i se generează cu operația **rfmk**:

$$K_i = rfmk(R, rfmk(R, T + i)),$$

unde T este ora curentă. Deoarece *rfmk* este funcție de *KMO*, rezultă că K_i este funcția de R, T și *KMO*.

Cheile de cifrare a mesajelor sînt generate în interiorul sistemului, utilizînd operația *rtmk*. Numele aleatoare R_i $(i = 1, 2, ...)$ sînt generate pornind de la două valori independente U_{i-1} și Z_i astfel:

$$R_i = rtmk(U_i, U_i),$$

unde

$$U_i = rtmk(U_{i-1}, Z_i), \quad \text{pentru } i > 0,$$

fiecare Z_i fiind derivat din două ceasuri independente. Valoarea inițială U_0 este obținută dintr-o serie de informații proprii utilizatorului și procesului. Fiecare U_i este funcție de *KMO* și toate precedentele valori ale lui U și Z, iar oricare dintre R_i depinde de *KMO* și U_i. Numerele aleatorii R_i sînt definite ca fiind valoarea cifrată a cheii K_i sub o altă cheie X:

$$R_i = E_X(K_i).$$

Cheile folosite în comun de un grup de utilizatori ai unui sistem teleinformatic pot fi generate pe baza unor informații proprii grupului. Fie G un grup de utilizatori într-o rețea cu N utilizatori. Membrii lui G folosesc în comun o cheie secretă a grupului K_G, care le permite să emită și să recepționeze mesaje confidențiale la alți membrii ai grupului sau să aibă acces la fișiere ce aparțin grupului. Denning descrie metode de derivare a tuturor

cheilor de grup posibile dintr-o listă cu N valori ale utilizatorilor. O metodă simplă calculează cheia K_G a grupului G de m utilizatori astfel:

$$K_1,\ K_2,\ \ldots,\ K_m$$

$$K_G = 2^{K_1 K_2 \ldots K_m} \bmod p,$$

unde $K_1, K_2, \ldots, K_m$ sînt cheile private ale utilizatorilor. Un utilizator care nu face parte din G nu poate determina pe K_G sau oricare din K_i ale membrilor lui G fără a calcula un logaritm discret. S-a arătat însă că această operație este nefazabilă dacă p are lungime mare (200 de cifre zecimale). Dacă se cere o cheie mai scurtă de 200 de cifre, K_G poate fi comprimat prin cifrare cu un cifru secvențial cu reacție sau printr-o cifrare-bloc înlănțuită și reținînd doar biții din dreapta.

5.5.4. DISTRIBUȚIA CHEILOR DE SESIUNE ÎN REȚELE DESCHISE

Cînd doi utilizatori (procese) doresc să comunice într-o rețea, utilizînd un algoritm de cifrare convențional, ei trebuie să folosească în comun o *cheie de sesiune* (sau cheie de comunicații) secretă. Soluția este ca un administrator al cheilor din sistem să furnizeze aceste chei atunci cînd începe sesiunea de lucru. Dar utilizatorii au posibilitatea să găsească și alte chei (sigure) pentru a schimba aceste chei secrete. În continuare se va descrie cum pot fi implementate aceste posibilități. Va fi prezentată mai întîi schema de distribuție centralizată a cheilor de sesiune care utilizează o autoritate centrală, S. Protocolul, publicat de Needham și Schroeder [N-1], presupune că fiecare utilizator are o cheie privată înregistrată la S. Dacă doi utilizatori doresc să comunice, unul dintre ei obține cheia de sesiune K de la S și o distribuie și celuilalt utilizator. O nouă cheie se va obține la fiecare sesiune, așa încît nici utilizatorii și nici S nu trebuie să păstreze lista cu cheile de sesiune. Utilizatorul A dobîndește o cheie, K, de la S pentru a o folosi în transmisiile cu un alt utilizator B, parcurgînd pașii descriși în continuare (fig. 5.30).

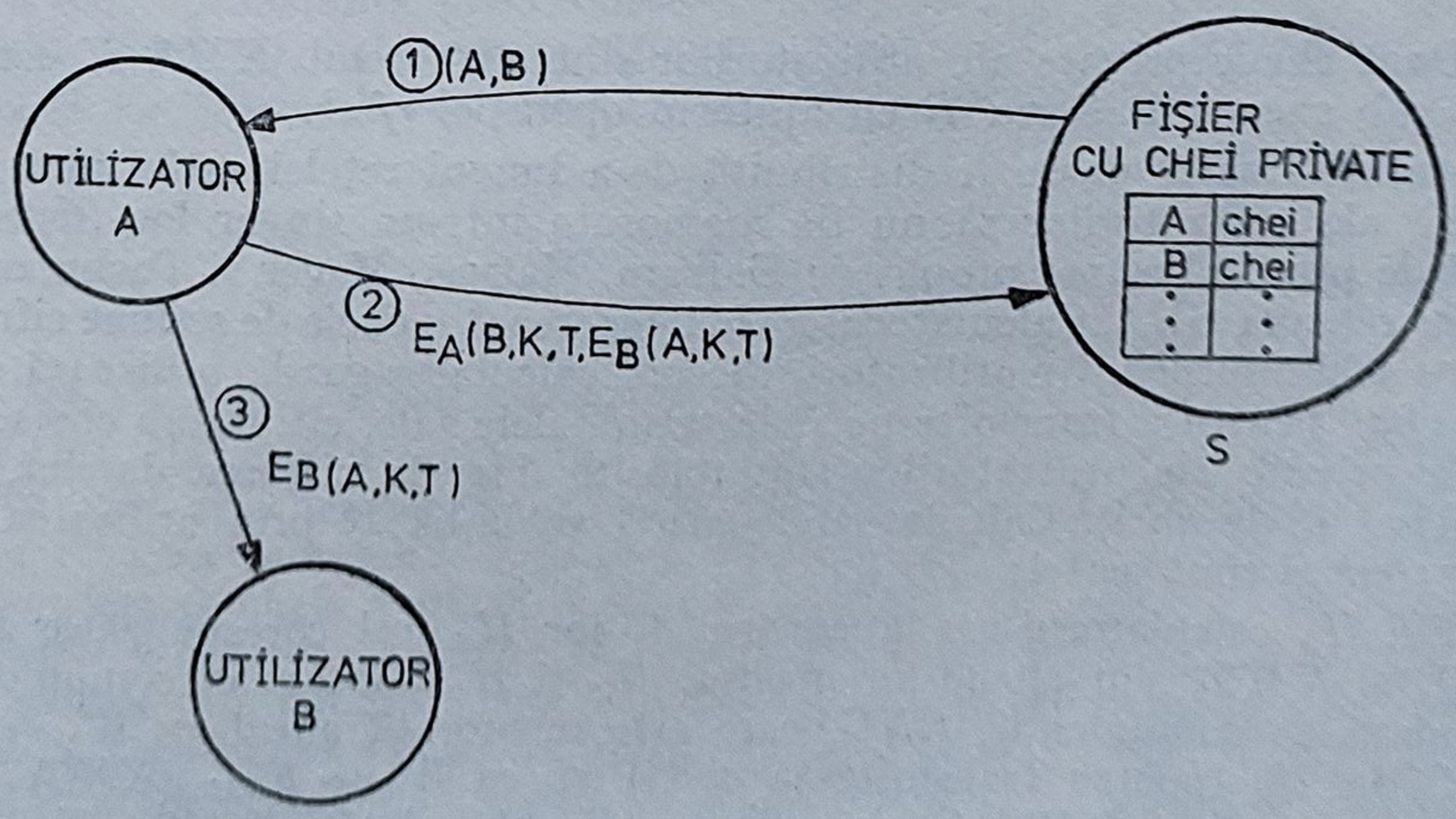


Fig. 5.30. **Protocolul de distribuție centralizată a cheilor**

• **Protocolul de distribuție centralizată a cheilor**

— A emite la S un mesaj clar:

$$(A, B)$$

conținînd identitatea lui A și B;

— S va emite către A mesajul cifrat:

$$E_A(B, K, T, C)$$

unde E_A este transformarea determinată de cheia privată a lui A, K este cheia de sesiune, T — data și timpul, iar

$$C = E_B(A, K, T)$$

conține cheia de sesiune cifrată sub cheia privată a lui B;

— A va emite lui B mesajul C de la pasul doi. În acest caz A și B dețin cheia de sesiune K, B avînd și identitatea (A) a utilizatorului care vrea să comunice cu el.

Pentru un plus de protecție, protocolul poate fi extins cu încă doi pași:

— B va genera un identificator aleatoriu I și va emite la A

$$X = E_K(I);$$

— A va descifra X, îl obține pe I, il va incrementa $(I' = I + 1)$ și îl va returna lui B:

$$Y = E_K(I').$$

În acest fel canalul logic cifrat între A și B a fost verificat și poate începe transmisia propriu-zisă de date.

Pentru protecția cheii de sesiune K între momentul generării și cel al cifrării cu cheile private ale lui A și B, K poate fi emisă direct sub forma unui număr aleatoriu, R, care este posibil să fie definit ca formă cifrată a lui K sub KMO:

$$R = E_{KMO}(K).$$

Dacă cheile private ale utilizatorilor sînt cifrate sub $KM1$, K este recifrată sub cheia lui A sau B cu ajutorul operației *rfmk*.

Autoritatea S poate fi distribuită de-a lungul rețelei astfel încît cheile private ale utilizatorilor să nu fie memorate într-un singur loc (fișier). Un astfel de protocol a fost propus de Ehrsam, Matyas, Meyer și Tuchman [E-1] pentru rețeaua SNA. Calculatorul-gazdă generează o cheie de sesiune cifrată R, pentru schimburile între utilizatori. Fiecare calculator-gazdă utilizează o operație *rfmk* pentru a transmite pe R la terminalele sale, cifrat sub cheia KMT (cheile private ale utilizatorilor sînt folosite doar la cifrarea datelor, nu și la schimbul de chei). Calculatoarele-gazdă schimbă R prin utilizarea funcțiilor *rfmk* și *rtmk* astfel:

Fie H_A calculatorul-gazdă pentru A, iar H_B cel corespunzător lui B; KMO_A și KMO_B sînt cheile lor master. H_A și H_B folosesc în comun o cheie secundară de comunicații, KNC, care este memorată cifrat ca $W_1 = E_{KM1_A}(KNC)$ în H_A (pentru transmiterea cheii) și ca $W_B = E_{KM2_B}(KNC)$ în H_B (pentru recepția cheii). Cheia cifrată K este dată de $R = E_{KMO_A}(K)$.

H_A utilizează o instrucțiune *rfmk* pentru a transmite pe R la H_B, iar H_B va utiliza *rtmk* pentru recepția lui R. Protocolul este următorul:

- **Protocolul de distribuție a cheii într-o rețea cu calculatoare gazdă**

(1) H_A va genera $R = E_{KMO_A}(K)$ și emite la H_B:

$$rfmk(W_A, R) = E_{KNC}(R).$$

(2) H_B obține pe R calculînd:

$$rtmk(W_B, E_{KNC}(R)) = R.$$

(3) H_A va emite terminalului cu utilizatorul A:

$$Z_A = rfmk(Y_A, R) = E_{KMT_A}(K),$$

unde KMT_A este cheia master a terminalului A și $Y_A = E_{KM1_A}(KMT_A)$. A obține K prin descifrarea lui Z_A.

(4) În mod similar H_B va transmite terminalului B:

$$Z_B = rfmk(Y_B, R) = E_{KMT_B}(K).$$

Acesta va obține pe K prin descifrarea lui Z_B.

Diffie și Hellman [D-9] au propus un protocol de schimbare directă a cheilor de sesiune. Metoda se bazează pe dificultatea calculării logaritmilor discreți (modulo număr prim). Primul utilizator, A, va alege o valoare aleatorie x_A în intervalul $[o, p - 1]$, unde p este un număr prim, p fiind făcut public la toți utilizatorii. Cel de al doilea utilizator, B, va alege o valoare aleatoare x_B în intervalul $(0, p - 1)$. Apoi A va emite la B.

$$y_A = a^{x_A} \bmod p,$$

iar B va emite la A valoarea:

$$y_B = a^{x_B} \bmod p,$$

unde a este o constantă prestabilită. Pentru valori mari ale lui x_A și x_B (de exemplu 664 biți), este computațional imposibil să se determine x_A și x_B din y_A și y_B. Cheia de sesiune va fi calculată de A astfel:

$$\begin{aligned} K &= (y_B)^{x_A} \bmod p \\ &= (a^{x_B} \bmod p)^{x_A} \bmod p \\ &= a^{x_A x_B} \bmod p. \end{aligned}$$

B va calcula cheia de sesiune în mod similar.

Firma MITRE a dezvoltat un sistem demonstrativ care utilizează o distribuție cu chei publice pentru schimbul de chei și algoritmul DES pentru cifrarea mesajelor. Fiecare utilizator A obține o cheie privată x_A și înregistrează cheia publică $y_A = a^{x_A} \bmod p$ la un centru (calculator) de distribuție a cheilor. Sistemul MITRE furnizează trei moduri de operare cu DES pentru cifrarea mesajelor: modul bloc standard, modul cu reacție și modul cu înlănțuirea blocurilor. Pentru ultimele două moduri doi utilizatori A și B trebuie să schimbe un vector de inițializare I_{AB} și I_{BA} în plus față de

cheia K de seslne. I_{AB} este folosit pentru transmisiile de la A către B pe cînd I_{BA} pentru cele de la B la A. Stabilirea unui canal logic sigur între A și B se face astfel:

- **Schimbarea cheilor utilizînd distribuția cu chei publice**

(1) A obține pe y_B din fișierul public și calculează cheia

$$K_{AB} = (y_B)^{x_A} \bmod p.$$

A va genera aleatoriu valorile R și R_{AB} și va calcula cheia de sesiune K și vectorul de inițializare:

$$K = D_{AB}(R)$$

$$I_{AB} = D_K(R_{AB}),$$

unde D_{AB} este transformarea de descifrare DES cu cheia K_{AB}. A va emite la B mesajul în clar:

$$(A, R, R_{AB})$$

(2) B obține y_A și calculează $K_{AB} = (y_E)^{x_A} \bmod p$ iar apoi pe:

$$K = D_{AB}(R)$$

$$I_{AB} = D_K(R_{AB}).$$

B va modifica I_{AB} printr-o metodă predeterminată și va obține I'_{AB} pe care-l cifrează:

$$X = E_K(I'_{AB}).$$

B generează apoi valoarea R_{BA}, calculînd:

$$I_{BA} = D_K(R_{BA})$$

și emite la A mesajul:

$$(X, R_{BA})$$

(3) A descifrează X, $D_K(X) = I'_{AB}$. Acest lucru îi confirmă recepția lui I_{AB} de către B. Apoi A va calcula:

$$I_{BA} = D_K(R_{BA}).$$

A va modifica pe I_{BA} în I'_{BA} și va calcula:

$Y = E_K(I_{BA})$, pe care-l va transmite la B.

(4) B va descifra Y, obținînd $D_K(Y) = I'_{BA}$, ceea ce confirmă recepția lui I_{BA} de către A.

În continuare orice mesaj cifrat între A și B va utiliza cheie de sesiune K, care are o durată de folosire mai scurtă decît cheia K_{AB}.

Calculul cheilor publice y_A și y_B și a cheii private K_{AB} se face în cîmpul $GF(2^n)$, cu $n = 127$ și cu polinomul ireductibil $p(x) = x^{127} + x + 1$.

Sistemul MITRE furnizează atît un mod de distribuție cu cheie publică central cît și unul local. Prin distribuția cheii în mod central, utilizatorii obțin cheile publice direct de la centrul de distribuție, ori de cîte ori doresc

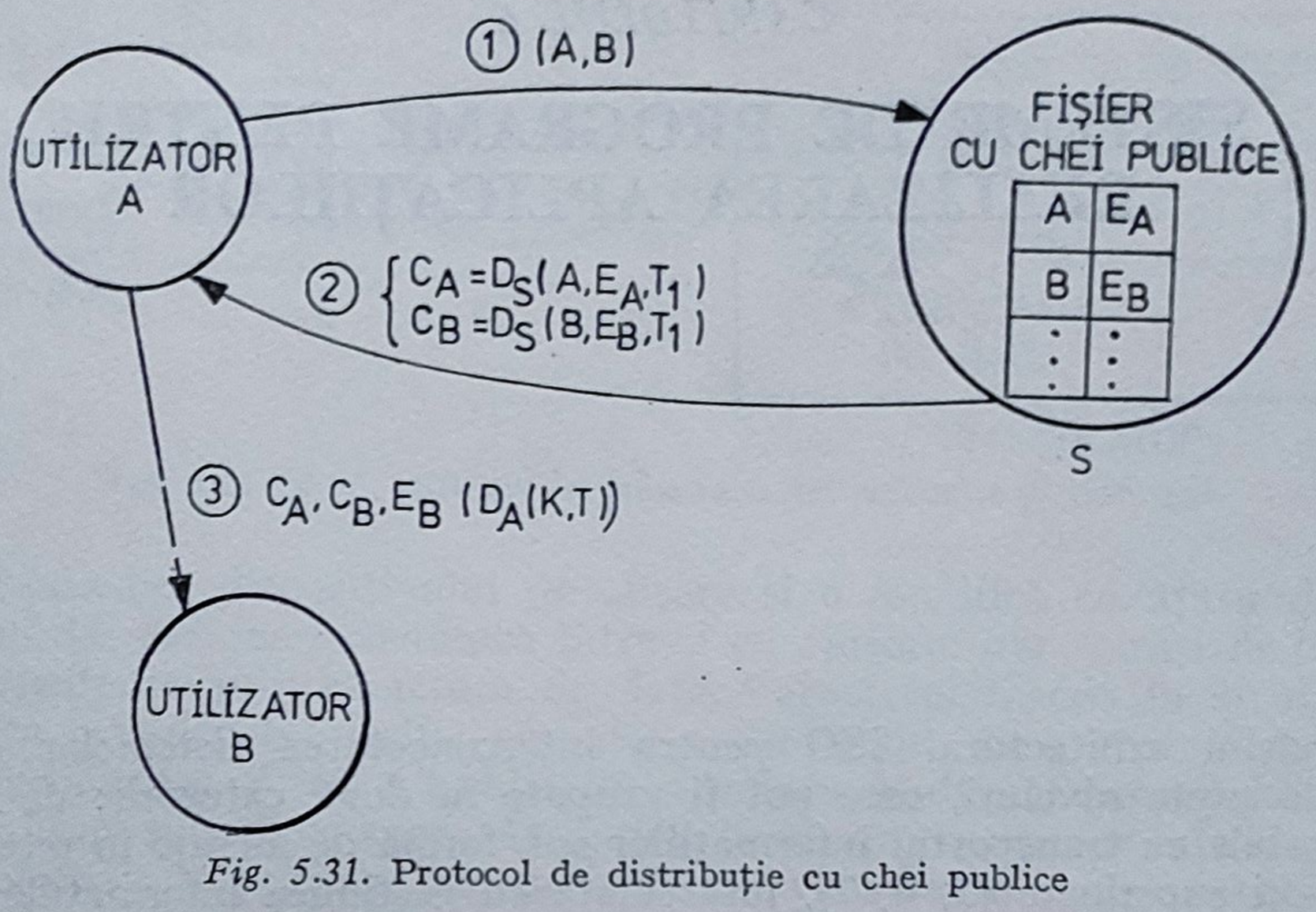


Fig. 5.31. Protocol de distribuție cu chei publice

stabilirea unor comunicații sigure. Prin distribuție locală, fișierul de chei publice este încărcat în sistemul utilizatorului și memorat la acesta.

Un criptosistem cu chei publice poate fi utilizat pentru schimbarea cheilor de sesiune. Conform protocolului cu certificate, fie C_A și C_B certificatele lui A și B, semnate de S. A și B pot schimba o cheie K cu următorul protocol (fig. 5.31).

- **Protocol de distribuție bazat pe chei publice**

(1) A emite mesajul de text-clar (A, B) la S, cerînd certificate pentru A și B.

(2) S emite la A certificatele

$$C_A = D_S(A, E_A, T_1)$$

$$C_B = D_S(B, E_B, T_1)$$

(3) A verifică C_A și obține transformarea publică E_B a lui B din C_B. Apoi A generează o cheie aleatoare K, timpul T și transmite la B:

$$(C_A, C_B, X)$$

unde $X = E_B(D_A(K, T))$, D_A fiind transformarea secretă de descifrare a lui A.

(4) B verifică C_B, obține transformarea publică E_A din C_A și calculează:

$$E_A(D_B(X)) = (K, T),$$

unde D_B este transformarea secretă a lui B. În acest moment atît A cît și B dețin cheia de sesiune K.

CAPITOLUL 6

SISTEME DE PROGRAME PENTRU REALIZAREA APLICAȚIILOR

Modelul arhitectural ISO pentru interconectarea sistemelor deschise cuprinde șapte niveluri, care pot fi grupate în două categorii: de bază — responsabile cu transportul informațiilor sub formă de mesaje în rețea (1, 2, 3, 4) — și superioare (5, 6, 7), însărcinate cu gestiunea informațiilor corespunzătoare sistemelor teleinformatice conectate. Spre deosebire de protocoalele de nivel inferior, a căror principală funcție constă în transmiterea datelor, procesele descrise în cadrul acestui capitol sînt responsabile cu gestiunea cerințelor impuse de utilizatorii finali. Cu toată diversitatea aplicațiilor solicitate de utilizatori unele dintre acestea, fiind mai frecvente, definesc funcții standard ale rețelei. Dintre ele menționăm protocoalele utilizate la manipularea terminalelor, transferul fișierelor, introducerea de la distanță a lucrărilor (la nivel prezentare), precum și protocoalele de mesagerie sau de interogare a unor baze de date distribuite (la nivel aplicație).

6.1. COMANDA TERMINALELOR. PROTOCOL DE TERMINAL VIRTUAL

În practică există o mare diversitate de terminale. Ori de cîte ori se dorește conectarea unui nou tip de terminal la calculator trebuie realizate module software de adaptare care să facă posibilă comunicarea. Ca atare cercetătorii au făcut studii pentru elaborarea unui terminal virtual (TV), cît mai universal posibil. În aceste condiții un calculator nu ar avea prevăzute decît comunicații cu un singur tip de terminal, iar adaptările la modelul fizic s-ar face prin serviciul de prezentare.

6.1.1. MODELUL TERMINALULUI VIRTUAL

Înainte de a se încerca definirea terminalului virtual și a protocolului acestuia va fi prezentat un model abstract pentru comunicația dintre terminale și aplicații [D-10] (fig. 6.1).

În cadrul modelului abstract considerat, terminalul și aplicația au acces la o structură de date bidimensională atît prin scriere, cît și prin citire. Aplicația are acces direct la structură, în timp ce utilizatorul citește structura

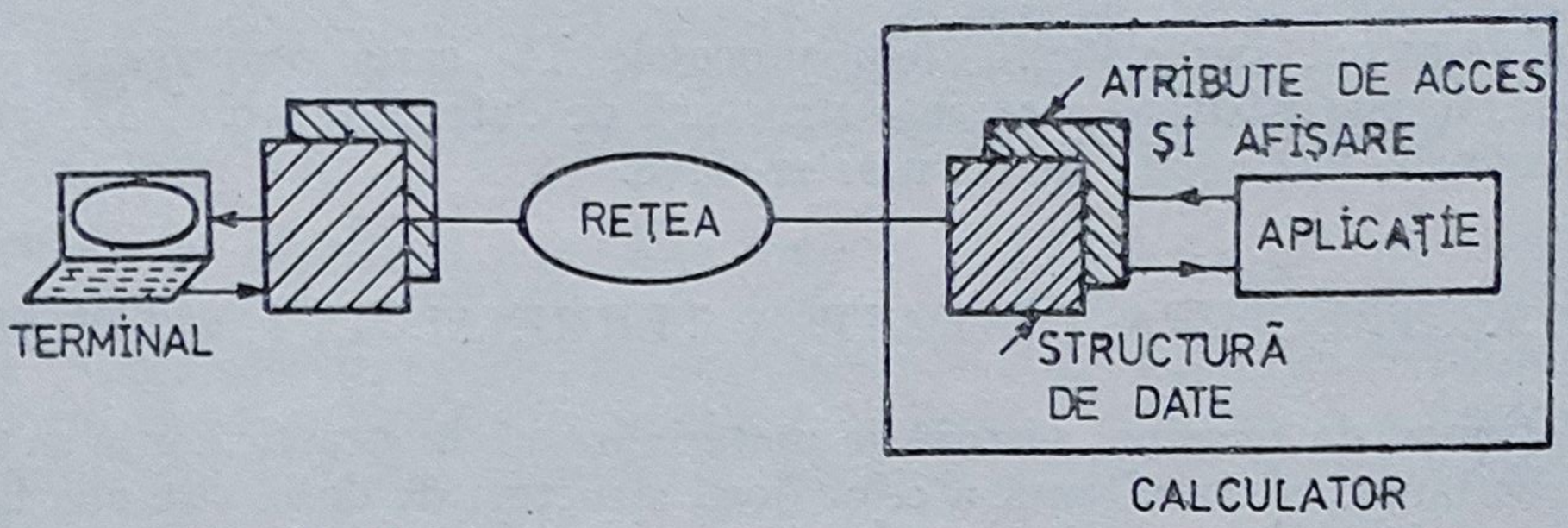


Fig. 6.1. Model abstract al comunicației aplicație — terminal

prin intermediul dispozitivului de afișare și o modifică cu ajutorul claviaturii. Cei doi parteneri comunică între ei cu ajutorul unor șiruri de caractere care tranzitează prin structura de date. Caracterul înscris în structura de date este introdus în poziție curentă a cursorului, care apoi avansează la poziția următoare. Fiecărei locații din structura de date îi este asociat un atribut de afișare și unul de acces. Atributele de afișare — așezate într-o structură bidimensională — definesc caracteristici de vizualizare (video invers, subliniere, clipitor etc.) iar cele de acces, zone din structura de date rezervate numai pentru citire sau scriere din punctul de vedere al unuia ori al altuia dintre parteneri. Structura de date se adaptează clasei de terminale din care face parte terminalul fizic; astfel, la terminalele cu afișare pe linie structura este unidimensională, iar la cele ce lucrează în mod paginat ea este bidimensională.

Terminalul virtual (TV) se definește ca un standard, în scopul manipulării uniforme a unei clase de terminale fizice. TV presupune existența atît la utilizator, cît și la aplicație a cîte unui proces — numit terminal virtual — care realizează *serviciul de terminal virtual*. La nivel arhitectural de prezentare serviciul permite cooperarea utilizatorilor, eliminînd incompatibilitățile dintre componentele unei clase de terminale. Datorită lui aplicațiile devin independente de tipul terminalului folosit. Cooperarea între procesele TV se face după un *protocol de terminal virtual* [P-2].

Un terminal virtual poate fi realizat ca un program alcătuind o parte din inteligența controlorului de terminale sau a sistemului de calcul. Există trei moduri de utilizare a conceptului de TV: pe legătura terminal-terminal, calculator-terminal ori calculator-calculator. În adaptarea modelului de TV nu trebuie să se pornească de la un set de componente virtuale corespunzătoare părților unui terminal real, evitîndu-se analogia cu terminalul real. Elementul central al modelului îl constituie structura de date, ca mijloc esențial de comunicație între terminal și calculator (fig. 6.2). Fiecare dintre aceștia schimbă sau consultă structura într-un mod convenabil pentru a transmite informația către celălalt: terminalul utilizează un protocol mod caracter, iar calculatorul — un protocol mod mesaj independent de primul. În cazul interacțiunilor

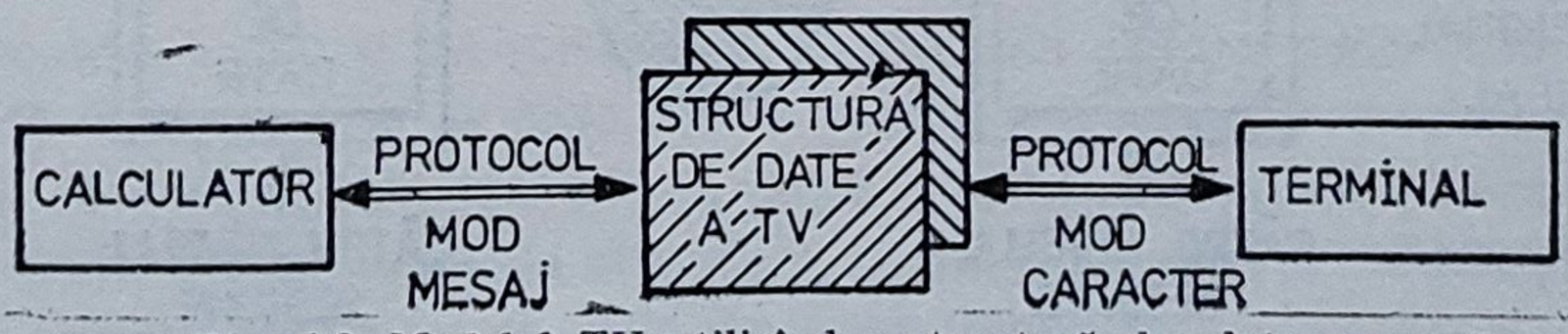


Fig. 6.2. Modelul TV utilizînd o structură de date comună

terminal-terminal sau sistem-sistem procesele TV care cooperează pot fi simetrice, fiecare păstrînd o copie a structurii de date. Protocolul este posibil să fie simetric, fără relații de tip master-slave.

6.1.2. STRUCTURA DE DATE A TERMINALULUI VIRTUAL

Noțiunea de terminal virtual se materializează în practică prin actualizarea combinată și simultantă a celor două structuri de date, cîte una la fiecare extremitate a comunicației. În figura 6.3 se prezintă un exemplu de structură de date în limbaj PASCAL ce materializează un terminal virtual simplu. Terminalul este reprezentat printr-o structură ce definește un ecran ca un ansamblu de cîmpuri formate din caractere. Există și alte informații necesare, cum ar fi poziția cursorului, tipul terminalului real conectat, tipul legăturii (duplex sau semiduplex).

Această structură de date este duplicată la cele două extremități ale comunicației, fiind manipulată de fiecare entitate de prezentare [P-3]. Fig. 6.4

```
type CÎMP = record
        TEXT: array [O ... Maxtext] of char;
        POS X: 0 ... Max X;
        POS Y: 0 ... Max Y;
        JUSTIF: (dreapta, stînga, între) end;

var TERMINAL = record
        ECRAN: array [O, Maxecran] of CÎMP;
        CURSOR: record
              x: 0 ... Maxtext;
              y: 0 ... Maxecran
              end;
        TIPTERM: (ARS 33, VTR 2);
        LEGĂTURĂ: (Full, Half);
        PAROLA: (eu, el);
        STARE: (Normal; Neinițializat)
        end;
```

Fig. 6.3. Structura de date a terminalului virtual

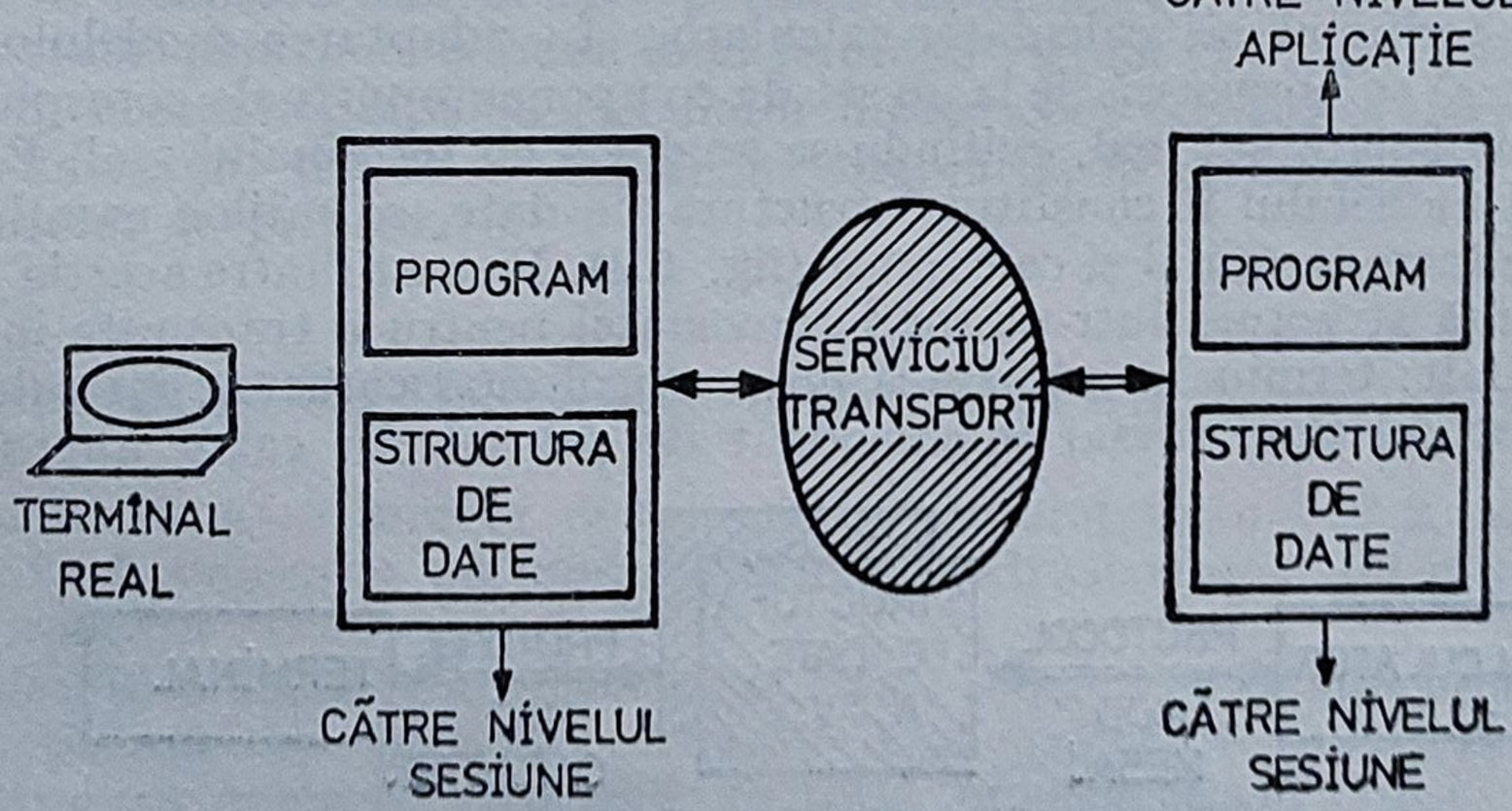


Fig. 6.4. Funcționarea protocolului de TV

ilustrează funcționarea paralelă a celor două servicii de prezentare. Se poate evidenția faptul că entitatea de prezentare din partea terminalului joacă un rol echivalent PAD-ului (unitate de asamblare/desasamblare pachete) într-o rețea X 25.

Odată cu stabilirea unei conexiuni la nivel prezentare are loc o fază de negociere între cei doi parteneri. Scopul ei este de a preciza limitele acceptabile (reale) pentru modelul general manipulat. În exemplul considerat este vorba de alegerea tipului de terminal, a tipului legăturii și a constantelor (în sens PASCAL) care intervin în definirea structurii. Starea terminalului va deveni deci „normală" dacă această negociere s-a terminat.

Structura de date permite ca la fiecare extremitate să se păstreze imaginea sigură a terminalului în curs de utilizare. Desigur, conceptul de structură de date nu este suficient. Normele în curs de elaborare prevăd, în plus față de structura de date (care e statică), prezența unor elemente dinamice ce pot fi asimilate cu programe. Programele conțin elementele de exploatare locală a terminalelor conectate.

Așa cum s-a arătat, în figura 6.1 structura de date este completată de o colecție de atribute (de acces și afișare) care asigură libertatea dorită în prezentarea informației. Dintre atributele tipice amintim:

— PROTEJAT — nu poate fi modificat de partener;
— NEAFIȘABIL — utilizat pentru citirea parolelor;
— ALINIAT DREAPTA — șirurile de caractere pot fi aliniate la dreapta;
— ALINIAT STÎNGA;
— TRANSMISIE INTERZISĂ — se poate afișa doar local;
— VIDEO INVERS/CLIPITOR — modul de afișare a cîmpului respectiv;
— DEFILARE/PAGINA — modul de scriere a cîmpului;
— COLORAT — afișarea cîmpului în culori;
— INTENSITATE — cîmpul afișat mai strălucitor;
— FOND — fondul cîmpului este colorat.

Numărul atributelor și modul lor de descriere reprezintă sarcini ale viitoarelor standardizări. Posibilitatea de a avea mai multe atribute opționale face necesară negocierea opțiunilor ce vor fi utilizate de cele două terminale virtuale înainte de a începe dialogul propriu-zis. Declararea și descrierea structurii de date sînt posibile printr-un limbaj apropiat de cel natural, tendință evidentă în unele implementări de terminal virtual.

Structurile de date sînt manipulate local prin protocoale care pot fi orientate mesaj sau orientate caracter. Sistemul local este responsabil de protocoalele locale și de rezolvarea conflictelor pentru accesul local și de la distanță la structurile de date. Unele studii prevăd ca Programul de terminal virtual să conțină cîte un proces pentru protocolul local, protocolul la distanță, controlul accesului la structura de date, precum și module de interpretare a comenzilor locale și de la distanță (fig. 6.5).

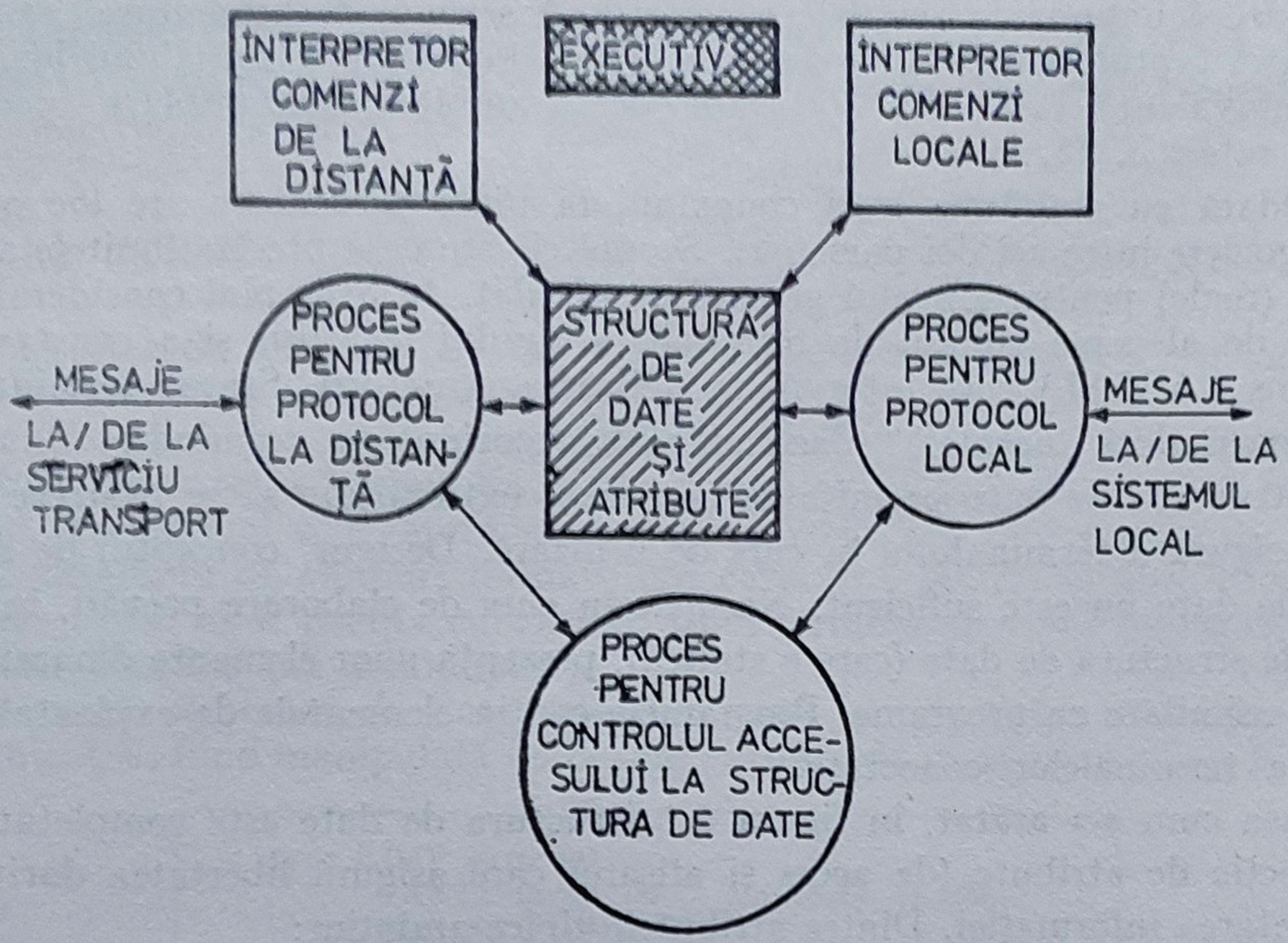


Fig. 6.5. Structura programului terminal virtual

TV va primi mesaje de intrare care vor avea ca efect modificarea structurii de date și generarea unor mesaje de ieșire ce raportează modificările de structură sau de conținut.

6.1.3. ANALIZA UNOR APLICAȚII

• *Terminalul virtual ARCHITEL,* aflat în fază de proiect de normă la nivel prezentare, este de tip videotext. Un astfel de terminal se compune din următoarele structuri: document, cursor, memorie de lucru, program de recepție, tabele.

Documentul este structura de bază care corespunde funcției principale a TV, și anume aceea de a prezenta documente, formulare etc. Cursorul permite să se desemneze, în fiecare moment, o dată elementară a documentului. Memoria de lucru și programul sînt asociate cu prelucrările locale ce trebuie efectuate. Programul de recepție este un fel de automat care reacționează cu ajutorul unor reguli de tipul: eveniment + condiție → acțiune. Un exemplu de eveniment poate fi acționarea unei anumite taste, sfîrșitul unui cîmp, recepția unui caracter eronat. O condiție este, de exemplu, cursorul la început de document sau la început de zonă de recepție. În fine, acțiunile pot fi emisia unor date la celălalt capăt al interacțiunii, un acces la memorie, etc.

Tabelele sînt de fapt definiții ale diverselor seturi de caractere ce vor fi utilizate în transmisie.

În afara structurilor de date, TV conține și valorile implicite ale fiecărei structuri, numite în proiectul ARCHITEL model de document, model de cursor etc. În figura 6.6 se prezintă modelul complet al TV ARCHITEL. Uneori acest model se numește profil și definește valorile implicite care carac-

Model de terminal virtual
Terminal virtual
    Model de document
    Document
        Model de pagină
        Pagină
            Model de zonă
            Zonă
                Model caractere
                Caractere
Model de cursor
Cursor
Model de memorie de lucru
Memorie de lucru
Model de program de recepție
Program de recepție
    Model de subprogram de recepție
    Subprogram de recepție

Fig. 6.6. Structura de date a TV videotext

terizează un anumit terminal. Se poate remarca faptul că anumite structuri se descompun în entități mai fine; de exemplu, un document se compune din pagini, care la rîndul lor se compun din zone, etc. Aceste structuri sînt exploatate de către entitatea de prezentare și au ca sarcină să creeze, în funcție de cerințele aplicației, date conforme cu structurile. Astfel, dacă un terminal este activat se creează o dată de tip TV care are valorile inițiale definite de profil. Ori de cîte ori se obține o nouă dată rezultă și fiecare din datele înglobate de aceasta, cu valori implicite. De exemplu, cînd se creează o nouă pagină, sînt obținute atîtea zone cîte sînt prevăzute în modelul de pagină. Valorile inițiale se pot modifica, de exemplu, pentru a adăuga o nouă zonă la o pagină existentă.

În termeni de protocol, mesajele schimbate între entitățile de prezentare permit manipularea explicită a fiecăreia dintre aceste date. Există o familie de mesaje de tip CREARE-X (X este document, pagină, zonă, cursor etc.) care facilitează crearea unei date conform cu modelul. Datele pot fi modificate printr-o serie de mesaje de tip MODIFICA-X. Fiecare dintre aceste mesaje posedă parametrii ce precizează exact acțiunile lor.

- **Modelul NAPLPS**

NAPLPS (North American Presentation Level Protocol Syntax) reprezintă o propunere de standard mondial la nivel de prezentare. El a fost conceput pentru dirijarea pe linii de comunicații de joasă viteză a informațiilor cu caracter grafic. Sistemul de codificare a informațiilor este compatibil cu codul ASCII. În cadrul acestui standard imaginile sînt descrise într-un sistem de coordonate virtuale cu ajutorul unor instrucțiuni de definire a imaginii. Din necesități de compatibilitate cu un număr mare de sisteme existente, NAPLPS permite, de asemenea, gestiunea unor seturi de caractere în maniera videotextelor.

La baza sistemului stă principiul conform căruia informațiile grafice sînt reprezentate sub formă de puncte, linii, arce sau poligoane într-un sistem de coordonate carteziene. Pentru NAPLPS imaginea este un dreptunghi

al cărui vîrf inferior stînga are coordonatele (0, 0), iar cel superior dreapta — coordonatele (1, 1). Avantajul unui astfel de sistem rezidă în independența totală față de echipamentul existent în ceea ce privește descrierea unui document. Fiecare implementare are sarcina de a realiza corespondența între acest format al documentului și definiția reală a imaginii sale. Apar însă și unele dificultăți; un ecran de televizor obișnuit nu este un pătrat perfect, ceea ce înseamnă că pe el nu se poate reprezenta decît aproximativ 3/4 dintr-o imagine virtuală NAPLPS. O altă dificultate constă în găsirea algoritmului care stabilește corespondența între imaginea NAPLPS și punctele unui ecran informatic (cu o rezoluție mai slabă).

În figura 6.7 este dat un exemplu simplu de specificare a unui desen. Fiecare comandă este codificată printr-un byte, urmat de informațiile de deplasare necesare. Numerele din cadrul comenzilor sînt coordonate absolute dacă nu au semn sau relative cînd sînt precedate de semn.

Codificarea informațiilor NAPLPS se face în ASCII. Fiecare caracter ori comandă pur grafică se reprezintă într-o tabelă de 256 de valori (codificare pe octet). Tabela este formată din 16 linii și 16 coloane și este împărțită în patru zone (fig. 6.8). În interiorul acestor zone se pot încărca sau se pot citi diferite seturi de caractere sau de comenzi. Astfel se realizează așa-numita extensie de cod. Într-un anumit moment sînt recunoscute și interpretate numai 256 de coduri. Se pot alege seturile GL (Grafic Left) și GR (Grafic Right) dintre următoarele șase disponibile: setul primar (codul ASCII); setul complementar (care regrupează caracterele utile în limbile străine și cîteva caractere grafice); setul de descriere al imaginilor (PDI-Picture Description Instructions; exemplul din fig. 6.7); setul mozaic (semigrafic); setul de caractere redefinibile dinamic (permit utilizatorilor să-și creeze propriul lor set de caractere); setul macro facilitează desemnarea prelucrărilor complexe preînregistrate printr-o singură comandă.

Seturile de control CO și Cl sînt fixe. Cu ajutorul lor se gestionează schimbarea seturilor G.

Aplicațiile ARCHITEL și NAPLPS reprezintă două contribuții interesante la normele ISØ [I-3] la nivel prezentare, în curs de elaborare. Ele își au originea în aplicațiile actuale de videotext și de grafică. Este foarte posibil ca unificarea acestor propuneri să survină odată cu publicarea definitivă a normelor.

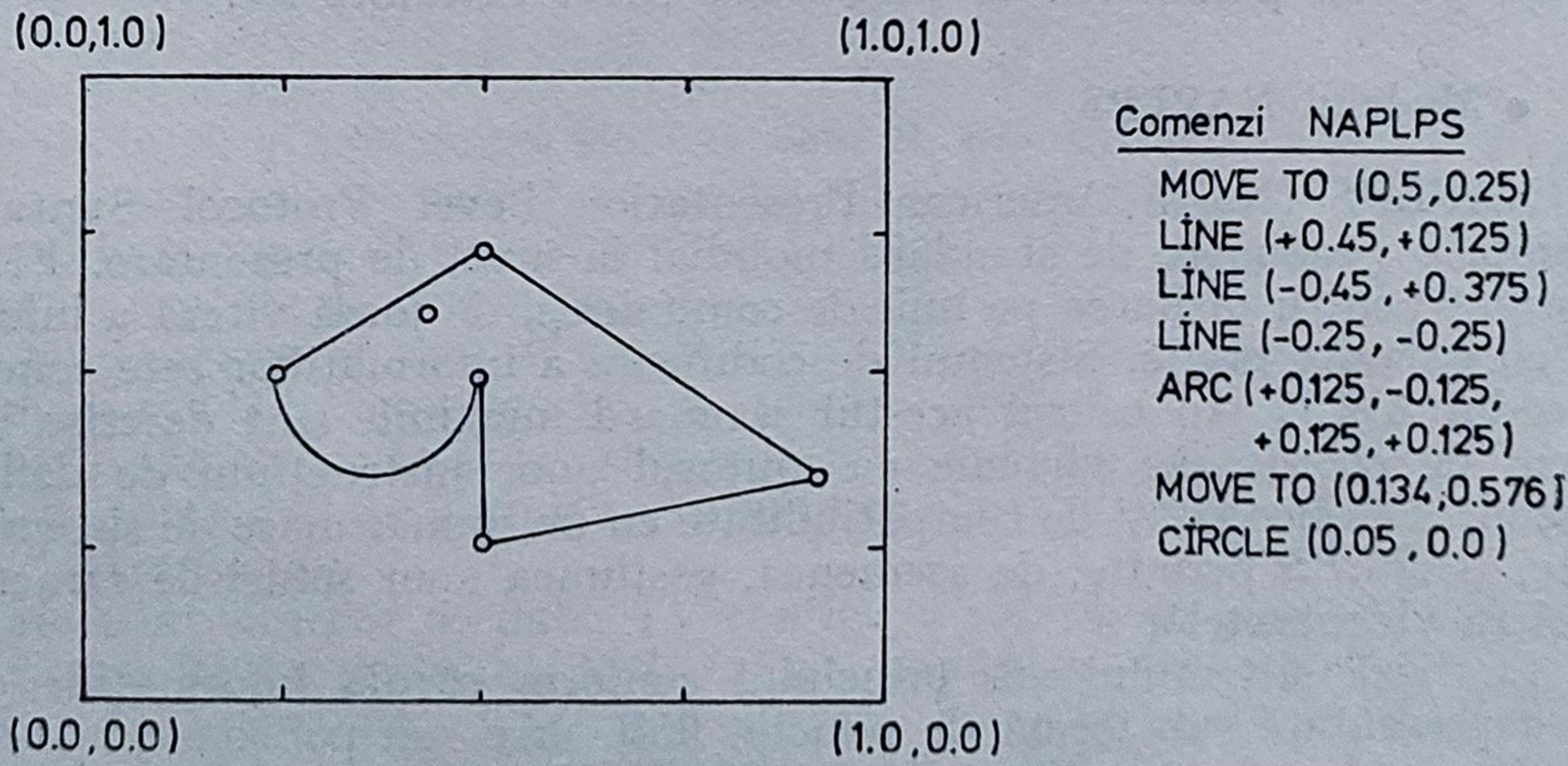


Fig. 6.7. Exemplu de desen NAPLPS

6.2. TRANSFERUL ȘI PRELUCRAREA FIȘIERELOR ÎN STI

Alături de accesul la resursele de prelucrare de la terminale interactive, alte funcții importante ale unei rețele distribuite de calculatoare sînt transferul și manipularea fișierelor. Componentă de bază a sistemelor de operare locale gestiunea fișierelor se realizează încă conform unor convenții proprii fiecărui tip de sistem în parte. Modul diferit în care sînt organizate sistemele de gestiune a fișierelor pe diverse calculatoare face dificil, dacă nu chiar imposibil accesul la fișiere și transferul lor între sisteme conectate în cadrul unor rețele. Acest impediment a condus la ideea construirii unui model de descriere a fișierelor și a atributelor lor care să permită definirea de servicii și protocoale pentru transferul de informații din fișiere, accesul de la distanță la datele sau la informațiile de control ale fișierelor. Un astfel de model reprezintă un *sistem de fișier virtual* (FV). Ca și în cazul terminalului virtual, rolul nivelului prezentare din modelul ISØ al interconectării sistemelor deschise este de a realiza traducerea (adaptarea) între structurile reale ale fiecărui calculator și această structură virtuală uniformă.

Organismele internaționale nu au definitivat standardizarea structurii de FV, așa încît elementele pe care le vom prezenta sînt conforme cu unele realizări și contribuții teoretice cu influență asupra elaborării viitoarelor norme [I-4], [E-2].

Un FV se poate defini ca o entitate ce posedă următoarele elemente: un nume care să permită identificarea FV fără ambiguități; atribute descriptive ale FV (de exemplu, dimensiunea, data creării); atribute structurale ce definesc structura logică și dimensiunea datelor stocate; datele memorate în fișier.

Serviciile și protocoalele care decurg din necesitatea obținerii unui context coerent de manipulare a acestor fișiere sînt compuse din patru faze principale: recunoașterea interlocutorilor (identitatea și drepturile reciproce); recunoașterea fișierului; recunoașterea structurii informațiilor din fișier; declanșarea prelucrărilor propriu-zise ale fișierului. Prima fază nu este specifică prelucrării FV; ea se aplică în general la toate prelucrările distribuite. În schimb, faza a doua stabilește identitatea fișierului de prelucrat, distingîndu-l după nume și atribute. Cea de a treia fază corespunde deschiderii fișierului din sistemele tradiționale de gestiune. Ultima, desemnează lansarea prelucrărilor specificate.

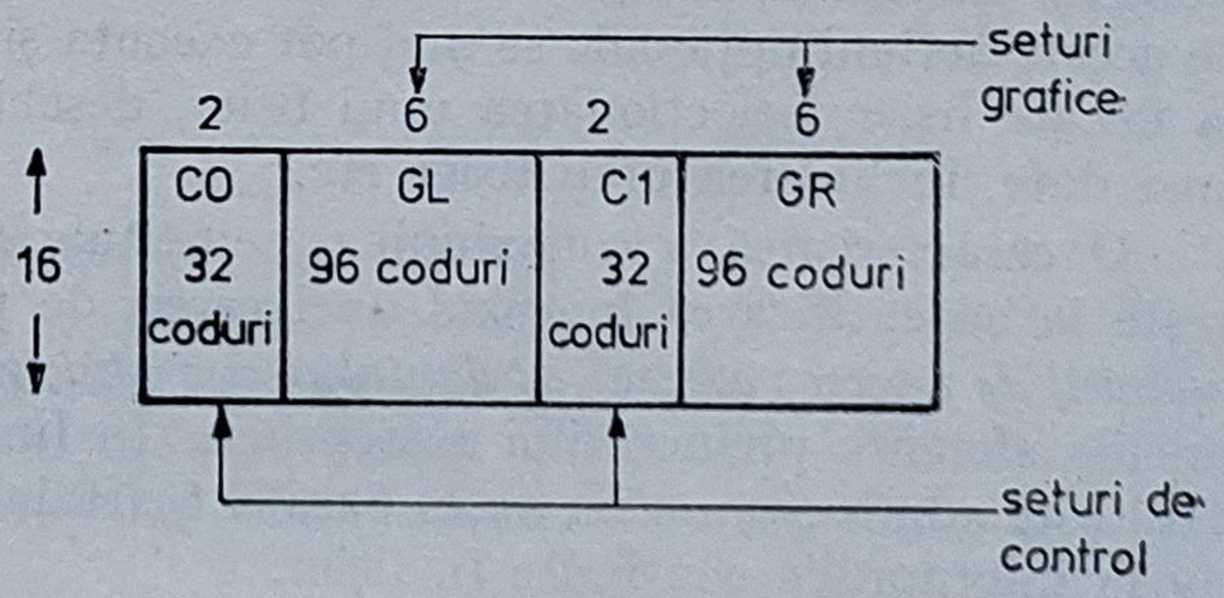


Fig. 6.8. Tabela de codificare NAPLPS

6.2.1. ATRIBUTELE DE CARACTERIZARE ALE UNUI FIȘIER VIRTUAL

Primul atribut de caracterizare a unui fișier este *numele*. Pentru a permite identificarea fără ambiguități a fișierului numele trebuie să fie unic. Structura numelor ce urmează să fie adoptată definitiv va fi cu siguranță una ierarhică care să permită reperarea ușoară a informațiilor pe un anumit calculator. Se prevede limitarea numelui unui fișier la 64 de caractere.

Al doilea atribut îl constituie *tipul de acces* posibil la un fișier: secvențial și aleator, excluzînd organizarea secvențial indexată. Deși foarte utilizată, organizarea indexată posedă prea puține caracteristici comune de la un calculator la altul pentru a putea face parte dintr-un model general. Odată ce organizarea a fost adoptată trebuie definit accesul posibil: citire, scriere scriere-citire. Se prevede fixarea acestui atribut la crearea fișierului, fără posibilitatea schimbării lui ulterioare.

Controlul accesului este reprezentat de o listă de condiții, dintre care cel puțin una trebuie să fie satisfăcută pentru a se permite accesul la informații. Controlul permite definirea operațiilor realizabile de către un utilizator (sau un grup) desemnat prin nume și eventual parolă. Dimensiunea parolei este prevăzută la 32 de octeți. Diferența dintre acest atribut și precedentul constă în restricția care se poate impune aici unor categorii de utilizatori în raport cu posibilitățile reale de prelucrare specificate de atributul precedent.

Contabilitatea este un atribut ce desemnează responsabilul (financiar) al prelucrărilor efectuate asupra informațiilor din fișier.

Informații cu privire la istoria fișierului se găsesc în conținutul altor atribute: data de creare, ultima lectură, ultima modificare, identitatea creatorului fișierului, identitatea utilizatorilor care au făcut ultimele modificări sau lecturi. Se prevăd și alte atribute privind tipul caracterelor utilizate (alfabetul de codificare), metoda de cifrare folosită etc. În fine dimensiunea fișierului actual, precum și cea care poate fi atinsă prin adăugări de informații vor fi exprimate în octeți.

6.2.2. ATRIBUTELE DE PRELUCRARE ALE UNUI FIȘIER VIRTUAL

Atributele de prelucrare caracterizează starea accesului la fișierul în curs. Primul atribut este *tipul de acces cerut* și poate fi: citire, inserare, înlocuire, ștergere, adăugare, stergere-fișier, citire-atribute, modificare-atribute. Față de aceste acțiuni opționale se mai pot executa și o serie de acțiuni obligatorii la fiecare fișier: selecționarea unui fișier, deschiderea unui fișier, localizarea unei date, închiderea unui fișier etc.

O caracteristică a conexiunii o constituie *identitatea apelantului*, ce servește la autentificarea în cazul unei cereri de prelucrare. Se mai utilizează *cuvîntul de trecere, adresa apelantului* și *contul*, atribute de care se leagă cheltuielile aferente prelucrărilor întreprinse. În fine, este prevăzut să se indice dacă este admis sau nu un acces paralel (partajat) ori, din contră, doar exclusiv la informațiile conținute în fișier.

6.2.3. SERVICIUL DE FIȘIER VIRTUAL

Serviciul de FV are rolul să creeze unele facilități de lucru prin care se pot atinge obiectivele cerute de inițiatorul sesiunii. Dialogul va permite: stabilirea reciprocă a identității celor două părți; identificarea fișierului necesar; precizarea atributelor care descriu fișierul și a volumului de date ce va fi transferat; declanșarea gestiunii fișierului la nodul unde este localizat; localizarea poziției în structura de acces a unităților de date ce vor fi accesate; transferul, înlocuirea sau ștergerea uneia sau a mai multor unități de date complete.

Utilizatorii serviciului FV se împart în două categorii: inițiatori (entitățile, aplicațiile care declanșează sesiunea) și receptori (entitățile aplicației ce manipulează FV). Serviciul de fișier virtual prezent într-o conexiune conferă un caracter intrinsec de asimetrie. Tipul prelucrărilor efectuate sînt, cel mai adesea, cerute de acea parte a conexiunii unde nu se află fișierul. Chiar și primitivele de fișier virtual posedă această caracteristică de asimetrie, în sensul că unele din ele nu pot fi emise de către ambii participanți. În figura 6.9 se dau numele diferitelor primitive disponibile în cadrul unui serviciu

Numele primitivei	Confirmare	Eșec	Apelantul primitivei
F. CONECT	X	0	→
F. RELEASE	X	–	→
F. ABORT	–	–	↔
F. SELECT	X	0	→
F. RECOVER	X	0	→
F. DESELECT	X	–	→
F. CREATE	X	0	→
F. DELETE	X	0	→
F. READ-ATTRIB	X	0	→
F. CHANGE-ATTRIB	X	0	→
F. OPEN	X	0	
F. CLOSE	X	–	→
F. READ	X	0	→
F. WRITE	X	0	→
F. DATA	–	F. CANCEL	↔
F. DATA-END	–	F. CANCEL	↔
F. TRANSFER-END	X	0	→
F. LOCATE	X	0	→
F. ERASE	X	0	→
F. CANCEL	X	–	↔
F. CHECK	X	F. CANCEL	↔
F. RESTART	X	F. CANCEL	↔
F. BEGIN-GROUP	X	–	→
F. END-GROUP	X	–	→

X = confirmare explicită
0 = furnizarea diagnosticului
→ = inițiatorul dialogului
↔ = ambii participanți

Fig. 6.9. Primitivele serviciului FIȘIER VIRTUAL

de fișier virtual, precum și tipul de răspuns pe care acestea îl cer în caz de succes (confirmare în caz de eșec).

În cadrul serviciului FV sînt definite următoarele regimuri de lucru, fiecare cu setul propriu de primitive:

• *Regimul de asociere la nivel aplicație,* care există atîta timp cît cele două aplicații asociate utilizează protocolul de FV. Principalele servicii aferente acestui regim sînt:

— *inițializare* (CONNECT), folosit de inițiator pentru a crea un regim de conectare la nivel aplicație între cele două servicii de FV;

— *terminare* (RELEASE), utilizat de inițiator spre a anula regimul de conectare;

— *terminare anormală* (ABORT), întrebuințat atît de inițiator, cît și de receptor la anularea necondiționată și înainte de termen a unei conectări;

• *Regimul de selectare a fișierului,* care identifică sau creează un fișier asupra căruia se va opera în fazele următoare. Acestui regim îi sînt asociate următoarele servicii:

— *selectare* (SELECT), utilizat de inițiator pentru a selecta un fișier existent și a-l pune la dispoziția funcțiilor de transport și acces la distanță;

— *deselectare* (DESELECT), folosit de inițiator la eliberarea fișierului selectat în prealabil prin SELECT;

— *creare* (CREATE), întrebuințat de inițiator spre a crea un fișier specificat și a-l selecta;

— *ștergere* (DELETE), utilizat de inițiator pentru a elibera un fișier selectat și a-l face să dispară.

• *Regimul de administrare fișier,* permite utilizatorului serviciului FV să citească sau să modifice atributele unui fișier selectat cu ajutorul funcțiilor *citire atribute* (READ-ATTRIB) și *modificare atribute* (CHANGE-ATTRIB);

• *Regimul de deschidere a fișierului,* ce pregătește fișierul pentru transferul unităților de date la care se face acces. Această fază pune la dispoziție următoarele servicii:

— *deschidere fișier* (OPEN), utilizat de inițiator la stabilirea contextului de prezentare și a controalelor asupra datelor transferate;

— *închidere fișier* (CLOSE), folosit de inițiator spre a elibera contextul stabilit prin OPEN;

• *Regimul de acces la conținutul fișierului* se realizează prin următoarele două servicii;

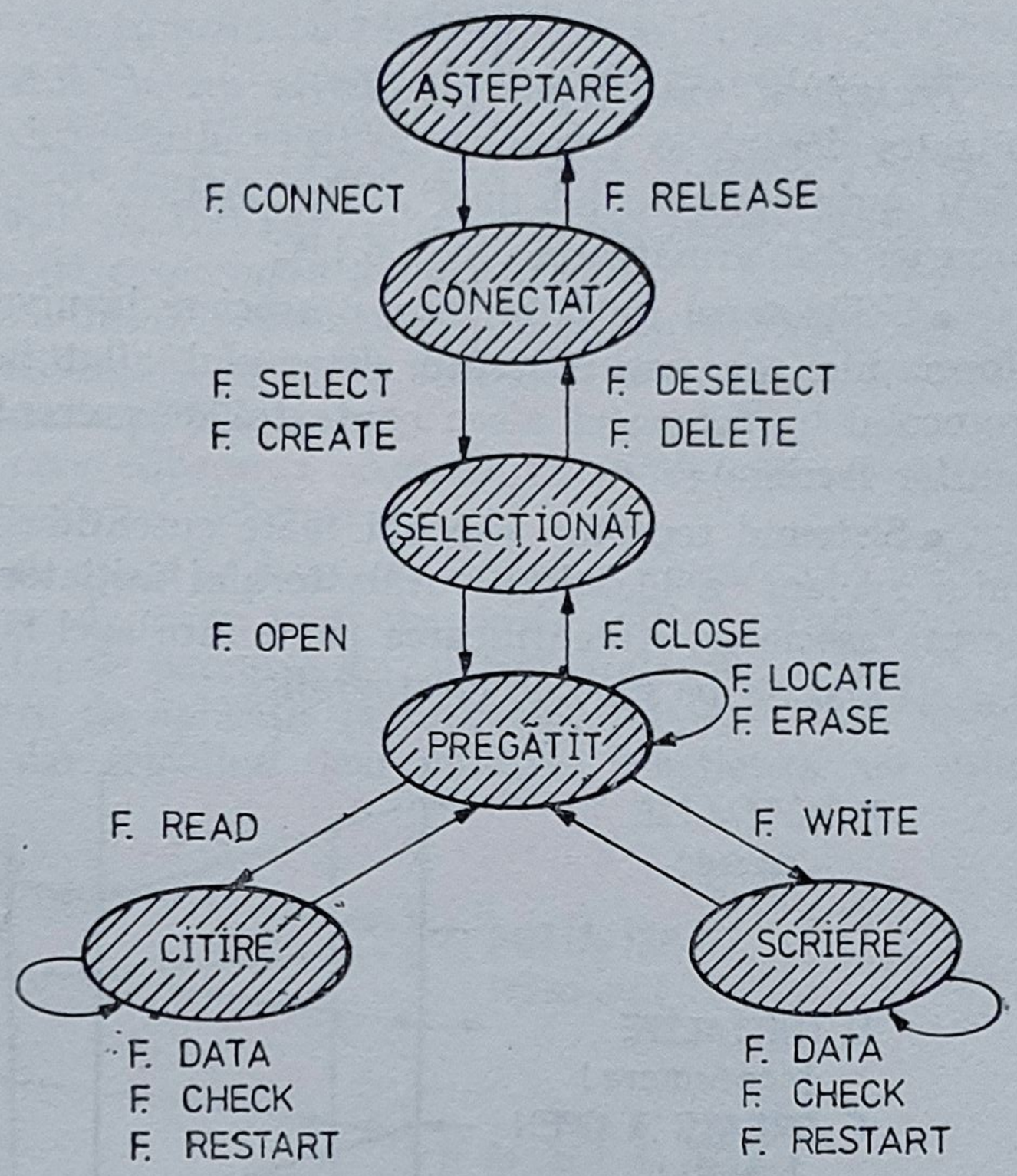
— *locatare* (LOCATE), întrebuințat de inițiator pentru a specifica identitatea unității de date la care se cere acces și care va trebui să fie localizată de către receptor;

— *ștergere* (ERASE), utilizată de inițiator, la scoaterea unei unități de date din fișier.

• *Regimul de transfer al datelor* constă într-un număr de operații de transfer și control al unităților de date. O unitate de date este, în general, formată dintr-o serie de elemente de date, dar acestea nu pot fi accesate sau actualizate în mod independent; prin urmare entitatea accesibilă este unitatea de date. Există șase servicii asociate cu transferul de date:

— *citire* (READ), folosită de inițiator la declanșarea unui transfer de date de la receptor (cu rol de emițător) către inițiator (cu rol de receptor de date);

Fig. 6.10. Diagrama de stare a serviciului de FIȘIER VIRTUAL



— *scriere* (WRITE), utilizat de inițiator pentru a declanșa o operație de transfer de la inițiator (cu rol de emițător) la receptor (cu rol de receptor de date);

— *transfer date* (DATA), întrebuințat de emițătorul de date la transmitere;

— *sfîrșit transfer date* (DATA-END), folosit de emițătorul de date spre a indica sfîrșitul transferului de date;

— *confirmarea sfîrșit de transfer* (TRANSFER-END), utilizat de inițiator pentru a confirma sfîrșitul transferului de date;

— *anularea transferului de date* (CANCEL), întrebuințat de ambele părți (emițător sau receptor de date) la anularea activității de transfer a datelor.

• *Regimul de reluare* are asociate două servicii;

— *marcare* (CHECK), utilizat de emițătorul de date pentru stabilirea unor puncte de reluare în fluxul de date;

— *restartare* (RESTART), folosit de ambele părți la întreruperea transferului în curs și negocierea unui punct de la care să se facă reluarea.

Înlănțuirea primitivelor prezentate dă naștere unei diagrame de stare simplificată, prezentată în figura 6.10. Ea prezintă activitatea unui proces de serviciu FV din punctul de vedere al comunicației.

6.2.4. PROTOCOALE CARE UTILIZEAZĂ FUNCȚIILE SERVICIULUI FV

În continuare vor fi prezentate spre exemplificare un număr de situații în care se folosesc funcțiile oferite de serviciul FV, indicîndu-se protocoalele cu ajutorul cărora au loc schimburile de date.

Transferul unui fișier complet la un alt sistem. Să presupunem că un utilizator dorește să transfere un fișier altui utilizator, situat pe un sistem diferit, prin intermediul unei rețele publice. Etapele pe care trebuie să le parcurgă sînt următoarele (fig. 6.11):

• Utilizatorul inițiator cere o asociere la nivel aplicație. Ea furnizează informațiile necesare adresării sistemului aflat la distanță și identificării. Protocolul implementat alege contextul de prezentare și selectează clasa de transfer (scriere);

• Sistemul receptor execută toate autentificările necesare asupra sistemului apelant, a identității utilizatorului inițiator și a contului acestuia și acceptă asocierea. Autentificarea poate face apel la procedurile de semnătură digitală bazate pe metode criptografice;

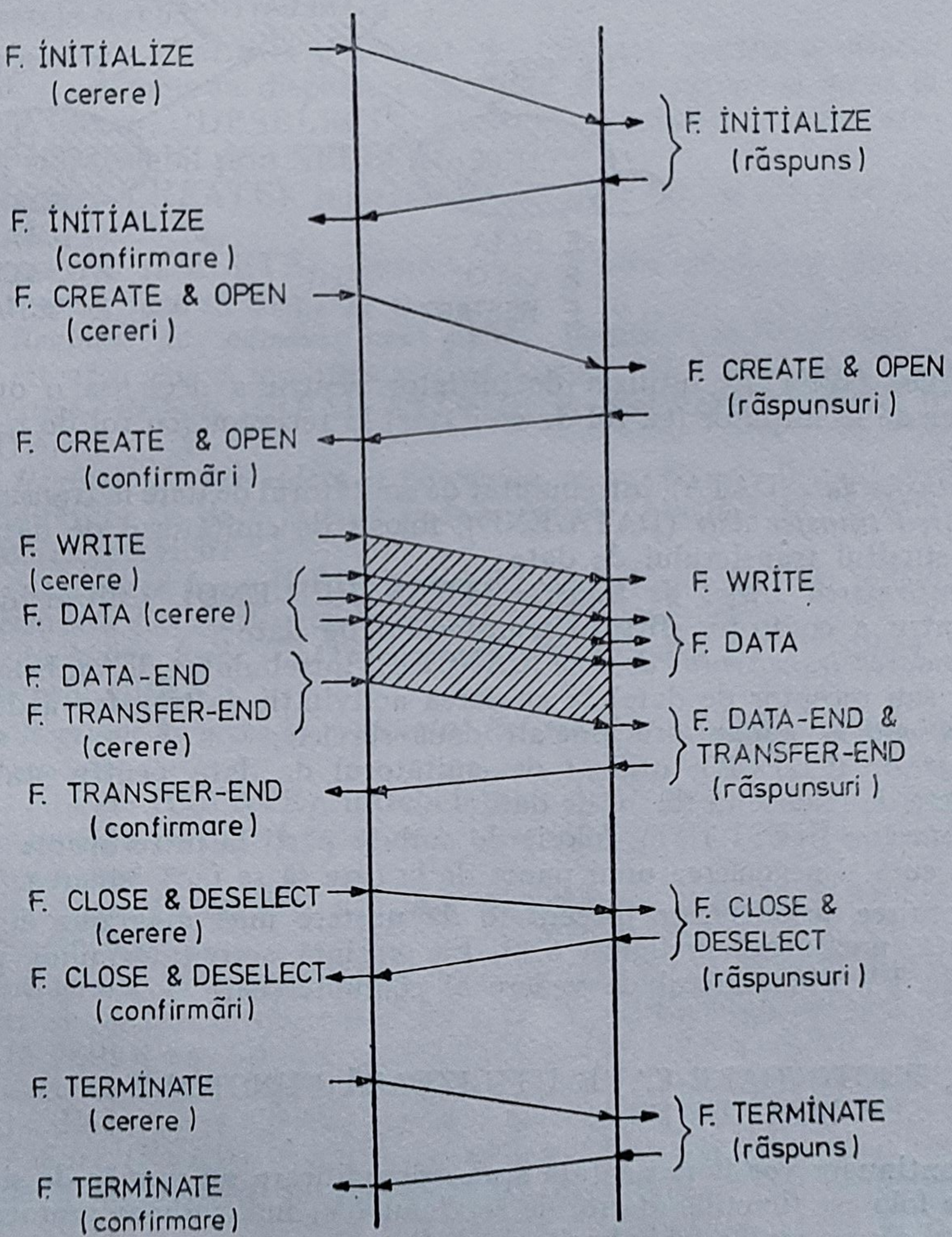


Fig. 6.11. Transferul unui fișier la un sistem aflat la distanță

• Sistemul inițiator emite specificațiile fișierului ce va fi creat pe sistemul aflat la distanță, precum și cele ale transferului de date care va fi executat;

• Sistemul receptor acceptă aceste specificații;

• Sistemul inițiator emite ca un șir continuu de transmisii, specificarea de execuție a unei operații de scriere, conținutul fișierului, markerul de sfîrșit de date și o cerere de confirmare a transferului;

• Sistemul receptor confirmă primirea conținutului fișierului;

• Sistemul inițiator eliberează legătura cu noul fișier creat;

• Sistemul receptor confirmă eliberarea;

• Sistemul inițiator cere eliberarea asocierii cu celălalt sistem;

• Sistemul receptor confirmă eliberarea.

Accesul la o bază de date aflată la distanță. Să presupunem că un utilizator dorește să transmită o serie de cereri unui sistem de gestiune a unei baze de date aflate la distanță. Etapele parcurse sînt următoarele (fig. 6.12):

— inițializarea asocierii de aplicație (asemănător primelor două operații din exemplul precedent, dar selectînd doar unitatea funcțională de citire);

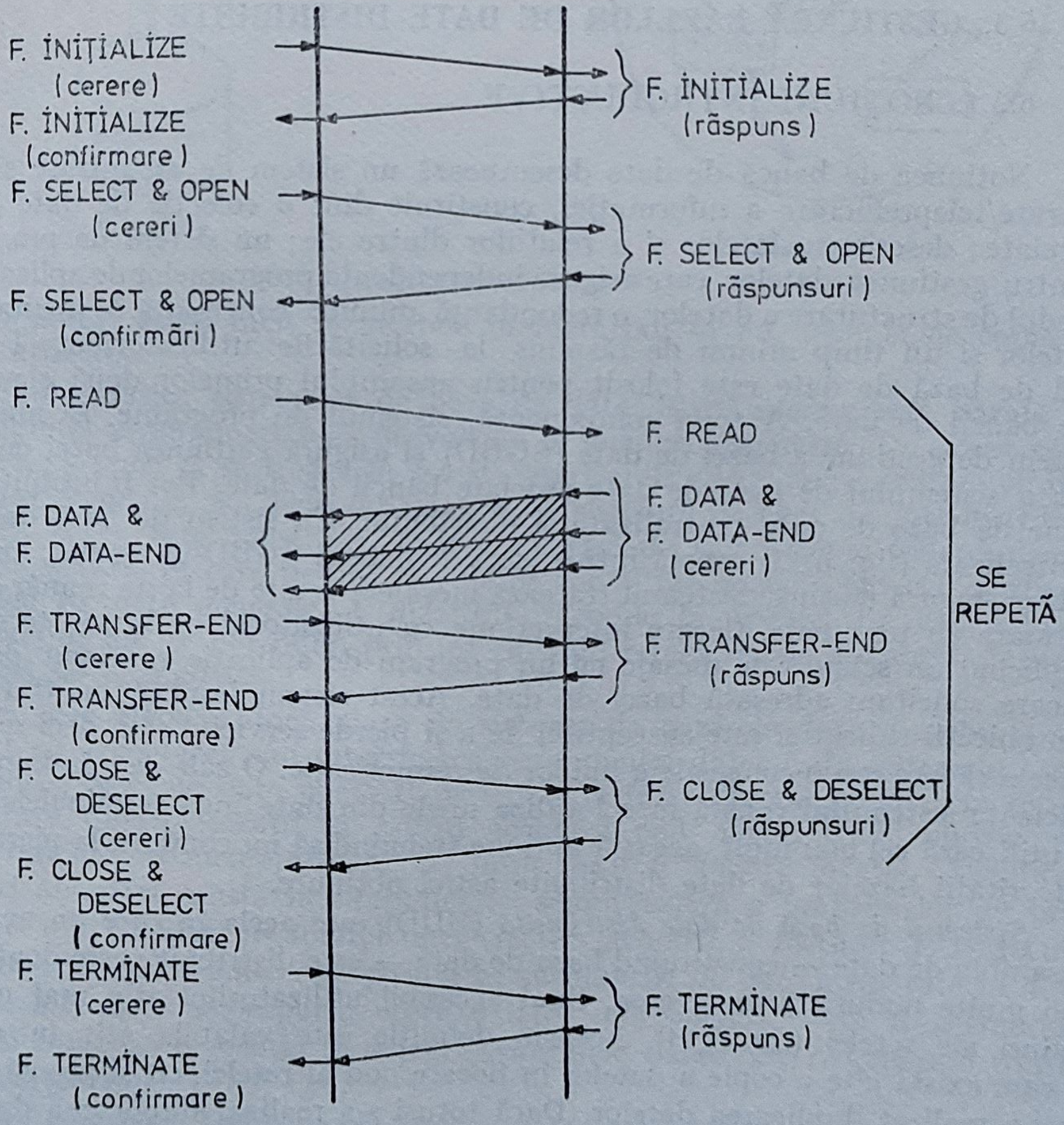


Fig. 6.12. Accesul la o bază de date

— inițializarea fișierului (la fel ca în pașii trei și patru din exemplul anterior. Însă în loc să furnizeze specificațiile unui fișier real, sistemul inițiator transmite un cîmp de informații cu ajutorul cărora se adresează baza de date);

— emiterea unei cereri de citire date de către sistemul inițiator, utilizînd o cerere la baza de date, ca și cînd ar fi identificatorul unei unități de date la care se solicită acces;

— returnarea rezultatului solicitării sistemului receptor; considerîndu-se că ar fi conținutul unității de date cerute;

— indicarea sfîrșitului tranzacției de către sistemul inițiator și confirmarea din partea receptorului;

— preluarea următoarelor cereri în mod asemănător;

— eliberarea legăturii de către sistemul inițiator și asocierea, care se efectuează la fel ca în etapele șase-nouă din exemplul precedent.

6.3. GESTIUNEA BAZELOR DE DATE DISTRIBUITE

6.3.1. NOȚIUNI INTRODUCTIVE

Noțiunea de bancă de date desemnează un sistem de organizare și prelucrare/teleprelucrare a informației, constituit din: o colecție de date intercorelate; descrierea datelor și a relațiilor dintre ele; un sistem de programe pentru gestiunea datelor, care asigură independența programelor de aplicație de modul de structurare a datelor, o redondanță minimă controlată în memorarea datelor și un timp minim de răspuns la solicitările utilizatorilor. Termenul de bază de date este folosit pentru ansamblul primelor două elemente ale băncii de date. A treia componentă, sistemul de programe, se numește sistem de gestiune a bazei de date (SGBD) și asigură gestiunea bazei de date sau a sistemului de baze de date asociate băncii de date. Pot fi întîlnite sisteme de baze de date centralizate și distribuite. Un sistem de bază de date centralizată (fig. 6.13) constă într-o bază de date și SGBD-ul asociat, ambele într-o singură mașină. Sistemul tratează mesajele venite de la un număr oricît de mare de terminale, fiecare interacțiune cu utilizatorul de la un terminal implicînd un schimb de mesaje cu un program de aplicație conceput pentru fiecare solicitare adresată bazei de date. Acest sistem implică un trafic de comunicații ridicat și este susceptibil de a-și pierde serviciul său, deci se produce o cădere sau o congestie a liniilor de comunicație. O cale de a îmbunătăți eficiența sistemului constă în a localiza unele din date lîngă terminalele care le utilizează cel mai mult, această opțiune trebuind să nu conducă la pierderea integrității bazelor de date distribuite astfel obținute.

Sistemul de bază de date distribuită (BDD) este acela în care un anumit ansamblu de date — constituind baza de date — este distribuit peste două sau mai multe noduri dintr-o rețea, fiind accesibil utilizatorilor din mai multe noduri ale rețelei (fig. 6.14). Această definiție este valabilă atît în cazul în care există cîte o copie a datelor în fiecare nod al rețelei, cît și atunci cînd nu s-a realizat duplicarea datelor. Dacă totuși s-a realizat duplicarea datelor din baza de date, toate copiile acesteia, din toate nodurile, trebuie actualizate

— inițializarea fișierului (la fel ca în pașii trei și patru din exemplul anterior. Însă în loc să furnizeze specificațiile unui fișier real, sistemul inițiator transmite un cîmp de informații cu ajutorul cărora se adresează baza de date);

— emiterea unei cereri de citire date de către sistemul inițiator, utilizînd o cerere la baza de date, ca și cînd ar fi identificatorul unei unități de date la care se solicită acces;

— returnarea rezultatului solicitării sistemului receptor; considerîndu-se că ar fi conținutul unității de date cerute;

— indicarea sfîrșitului tranzacției de către sistemul inițiator și confirmarea din partea receptorului;

— preluarea următoarelor cereri în mod asemănător;

— eliberarea legăturii de către sistemul inițiator și asocierea, care se efectuează la fel ca în etapele șase-nouă din exemplul precedent.

6.3. GESTIUNEA BAZELOR DE DATE DISTRIBUITE

6.3.1. NOȚIUNI INTRODUCTIVE

Noțiunea de bancă de date desemnează un sistem de organizare și prelucrare/teleprelucrare a informației, constituit din: o colecție de date intercorelate; descrierea datelor și a relațiilor dintre ele; un sistem de programe pentru gestiunea datelor, care asigură independența programelor de aplicație de modul de structurare a datelor, o redondanță minimă controlată în memorarea datelor și un timp minim de răspuns la solicitările utilizatorilor. Termenul de bază de date este folosit pentru ansamblul primelor două elemente ale băncii de date. A treia componentă, sistemul de programe, se numește sistem de gestiune a bazei de date (SGBD) și asigură gestiunea bazei de date sau a sistemului de baze de date asociate băncii de date. Pot fi întîlnite sisteme de baze de date centralizate și distribuite. Un sistem de bază de date centralizată (fig. 6.13) constă într-o bază de date și SGBD-ul asociat, ambele într-o singură mașină. Sistemul tratează mesajele venite de la un număr oricît de mare de terminale, fiecare interacțiune cu utilizatorul de la un terminal implicînd un schimb de mesaje cu un program de aplicație conceput pentru fiecare solicitare adresată bazei de date. Acest sistem implică un trafic de comunicații ridicat și este susceptibil de a-și pierde serviciul său, deci se produce o cădere sau o congestie a liniilor de comunicație. O cale de a îmbunătăți eficiența sistemului constă în a localiza unele din date lîngă terminalele care le utilizează cel mai mult, această opțiune trebuind să nu conducă la pierderea integrității bazelor de date distribuite astfel obținute.

Sistemul de bază de date distribuită (BDD) este acela în care un anumit ansamblu de date — constituind baza de date — este distribuit peste două sau mai multe noduri dintr-o rețea, fiind accesibil utilizatorilor din mai multe noduri ale rețelei (fig. 6.14). Această definiție este valabilă atît în cazul în care există cîte o copie a datelor în fiecare nod al rețelei, cît și atunci cînd nu s-a realizat duplicarea datelor. Dacă totuși s-a realizat duplicarea datelor din baza de date, toate copiile acesteia, din toate nodurile, trebuie actualizate

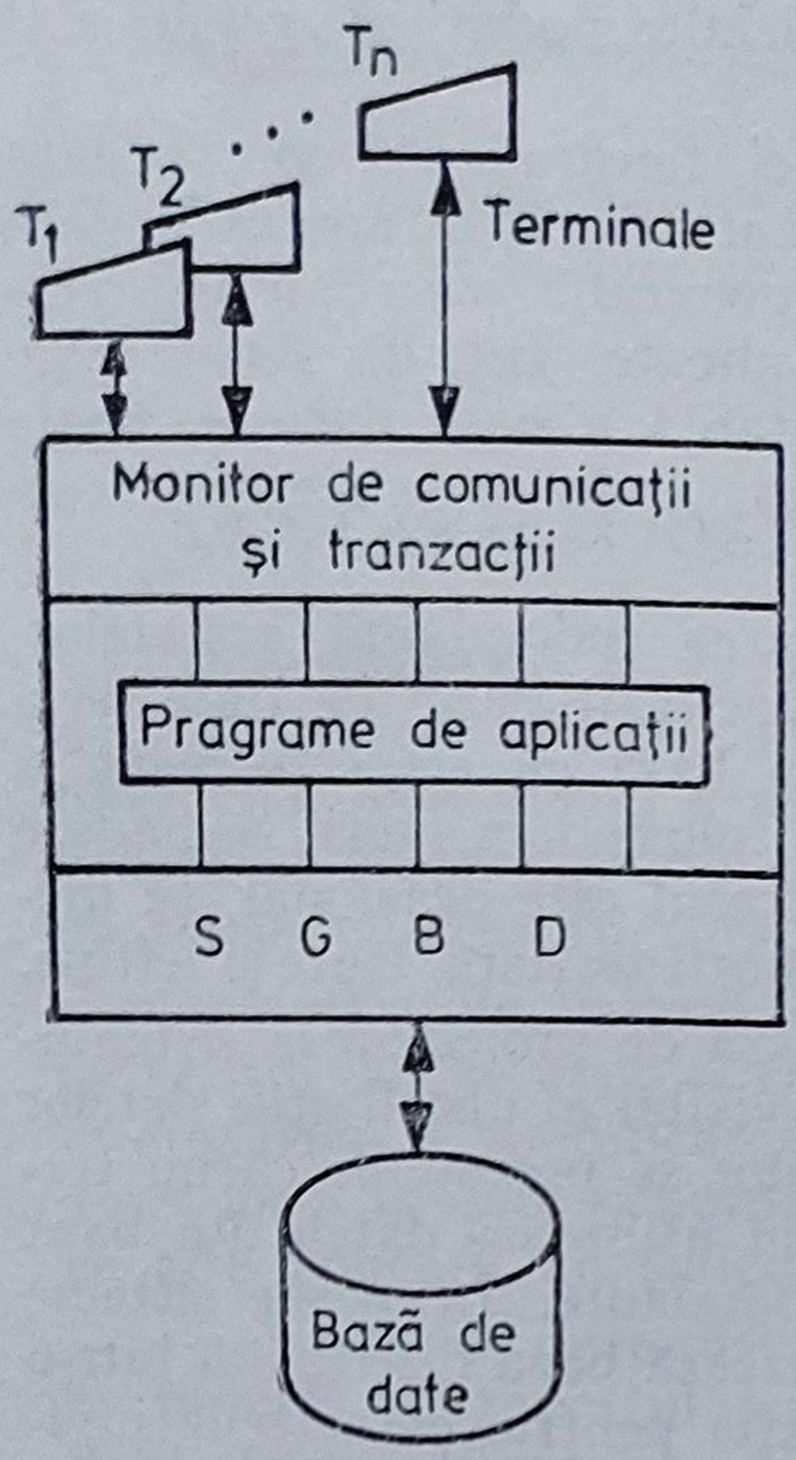


Fig. 6.13.

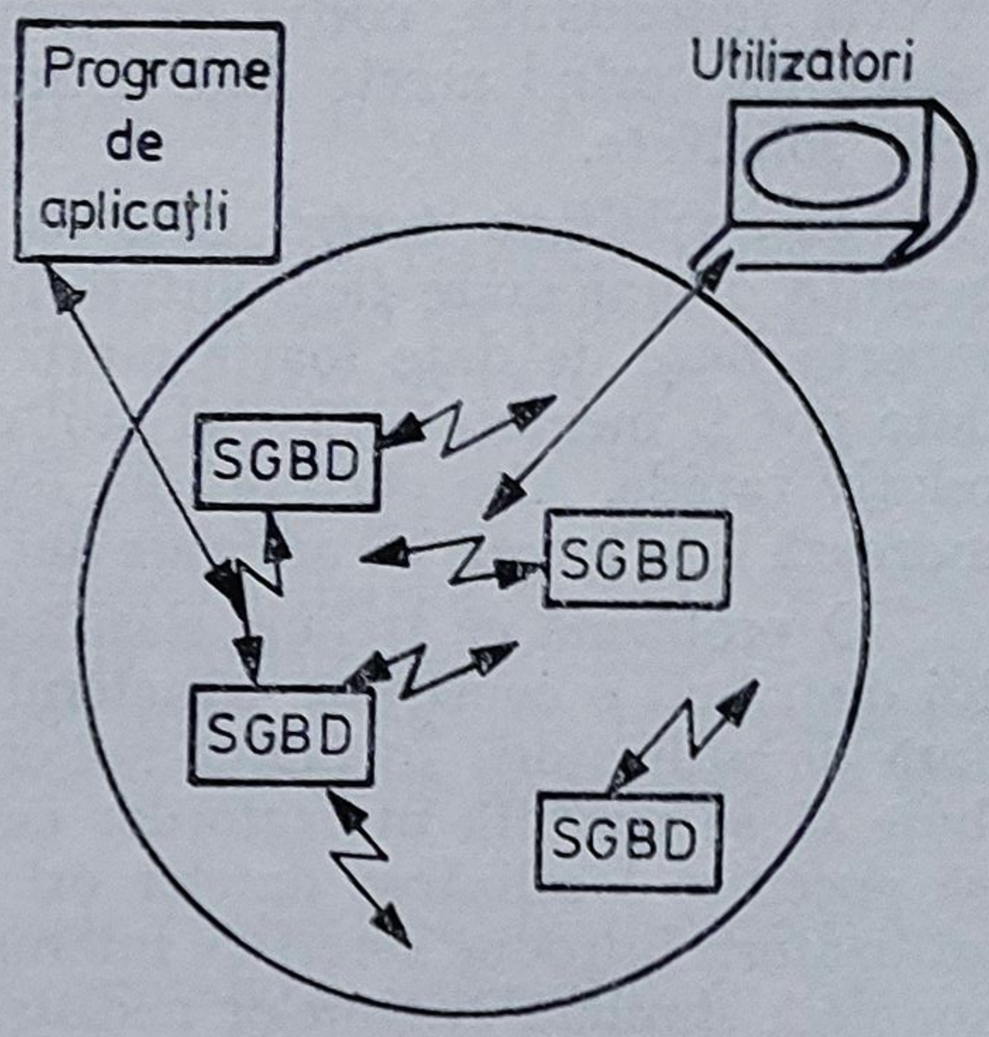


Fig. 6.14.

concomitent, astfel încît rezultatele prelucrărilor să nu depindă de copia de date utilizată.

Problematica sistemelor de baze de date distribuite nu este bazată pe concepte și presupuneri bine dezvoltate și unanim acceptate. De aceea se va încerca pe scurt să se treacă în revistă aserțiunile specifice pentru activitatea în acest domeniu. S-a acceptat unanim că sistemul de bază de date distribuită (fig. 6.14), este constituit dintr-o colecție de calculatoare independente, interconectate, în care fiecare calculator constituie suport pentru o porțiune a bazei de date generale. Utilizatorii introduc la oricare din calculatoare tranzacțiile lor, ce pot referi prin intermediul rețelei datele memorate într-un alt nod de la distanță.

Se pot distinge două clase de sisteme de baze de date distribuite, în raport cu acceptarea sau interzicerea redondanței datelor. În sistemele neredondante, baza de date poate fi partiționată între diverse noduri din rețea, sau între diferite locații în cadrul aceluiași nod, nici-o dată componentă neputînd însă fi memorată în mai mult decît o singură locație. Avantajul acestei abordări constă în simplificarea semnificativă a problemelor de proiectare a sistemului, pe fondul unei creșteri nedorite a costului de realizare. În sistemele redondante este posibil ca subseturile de date memorate în diferite locații să se suprapună, adică o dată componentă poate fi memorată în mai mult de o locație în rețea. Avantajele sistemelor redondante constă în următoarele:

• *Fiabilitate.* Dacă nu ar exista date redondante, căderea unui nod particular ce conține o bază de date, va cauza blocarea derulării aplicației

ce necesită datele memorate acolo, chiar dacă celelalte noduri ale rețelei sînt în stare de bună funcționare.

• *Eficiența comunicației.* În lipsa redondanței, fiecare dată componentă trebuie memorată într-un nod anume bine precizat. Dacă o anumită dată componentă este solicitată din locații dispersate geografic, va rezulta o creștere substanțială a costurilor de comunicație implicate. Într-un sistem care suportă redondanța, costul de comunicare la distanță a unor date se poate reduce memorînd aceste date în mai multe locuri, acolo unde ele sînt frecvent solicitate.

• *Flexibilitate pentru baza de date.* Memorarea redondantă a datelor, permite o mai mare flexibilitate în sporirea capacității bazei de date de a suporta baze de date foarte mari. Porțiunile din bază cu acces frecvent la date pot fi memorate în mai multe locații mici, folosind memorii secundare relativ rapide. Alte porțiuni din bază, la care accesul este ocazional, se memorează la o locație de arhivare din rețea, pe memorii terțiare, lente și ieftine.

O problemă ce trebuie clasificată încă din faza de proiectare a sistemului distribuit o constituie caracterul vizibil sau invizibil al distribuției datelor față de utilizatori. În marea majoritate a cazurilor se presupune că nu trebuie să se permită utilizatorilor care au acces sau utilizează datele din bază să specifice localizarea datelor ori a copiilor redondante. În aceste sisteme utilizatorii introduc cereri de interogare și actualizează baza distribuită într-o manieră similară sistemelor nedistribuite. Motivația pentru caracterul invizibil al distribuției datelor rezidă în realizarea independenței datelor din bază față de utilizatori, în vederea asigurării consistenței bazei de date — prin care toate copiile redondante ale bazei de date sînt aduse concomitent la zi, fiind identice în oricare nod din rețea s-ar găsi. Se obține astfel conservarea integrității datelor din baza distribuită.

Din punctul de vedere al performanțelor, rețeaua de calculatoare care suportă baza distribuită poate fi de următoarele tipuri:

• *rețea locală* de tip Ethernet, cu o lărgime mare de bandă de frecvență, întîrzieri de propagare reduse, potrivită pentru distanțe scurte;

• *rețea cu comutare de pachete* (precum ARPANET), caracterizată de o lărgime mai mică a benzii de frecvență, întîrzieri mai ridicate față de mediile de tip Ethernet, fiind utilizabilă pentru distanțe mari;

• *configurație de circuite închiriate punct-la-punct*; această rețea posedă o bandă de trecere și mai îngustă față de cazul anterior, dar are întîrzieri mai scăzute;

• *rețea radiodifuzată* prin transmisie radio clasică sau prin satelit, asigurînd o mare bandă de frecvență și întîrzieri de propagare de ordinul sutelor de milisecunde (rețea cu radiodifuzare prin satelit de tip ALOHA).

Aceste canale de comunicație acoperă trei ordine de mărimi relative la lărgimea de bandă și peste șase ordine de mărime în ceea ce privește întîrzierea de propagare.

Indicatorii de performanță menționați mai sus pentru rețelele de calculatoare au o influență semnificativă asupra proiectării sistemelor de baze de date distribuite susținute de aceste rețele, prin afectarea unor elemente esențiale precum:

• asigurarea propagării în timp real a informației de control către toate nodurile unde se găsesc baze de date;

• realizarea actualizării în timp real a tuturor copiilor de date redondate din bază;

• compromisul între costul transportului datelor din loc în loc ca urmare a cererilor de prelucrare complexă și costul realizării unor calcule suplimentare locale.

6.3.2. CARACTERISTICI ALE SGBDD DIN REȚELE PUBLICE

În continuare se investighează posibilitățile de realizare a bazelor de date distribuite în rețelele publice, definite în modelul de referință ISO-OSI, fiind apoi identificate o serie de lipsuri ale arhitecturii OSI din punctul de vedere al BDD. Tipurile de SGBDD care se folosesc în aplicații suportate de rețele publice au caracteristici distincte în trei direcții diferite:

— aplicațiile variază în funcție de dispersia datelor în baza de date și de gradul de dispersie al cererilor de acces la diferite locații, pentru execuția interogărilor și modificarea cererilor. Unele aplicații necesită în principal un acces local la date și aproape nici un acces la datele de la distanță, în timp ce alte aplicații au solicitări inverse;

— bazele de date distribuite din rețelele publice pot, de asemenea, varia în funcție de raportul omogenitate/eterogenitate a sistemelor de baze de date de pe diferitele calculatoare-gazdă conectate la rețea. Se deosebesc aplicații care rămîn total omogene, întrucît toate datele din bazele de date de interes pentru aceste aplicații, sînt manipulate de același SGBD. Alte aplicații pot fi răspîndite între mai multe baze de date care sînt manipulate de SGBD-uri diferite;

— modelul OSI a fost propus pentru a suporta comunicații între sisteme de calculatoare de diferite tipuri din rețeaua publică de date. Bazele de date distribuite din rețelele publice se disting între ele în funcție de reprezentarea internă a datelor, schemele de alocare a datelor și modalitățile de acces. Este important să se facă distincție între baze de date omogene și eterogene, suportate de sisteme de calculatoare similare sau diferite.

• *Caracteristicile sistemelor deschise*. Regimul de comunicații în modelul arhitectural a fost propus să suporte comunicația doar între doi parteneri ai rețelei. Acest tip de disciplină de comunicație, de tip punct-la-punct, semnifică în esență că regăsirea în baza de date sau cererile de modificare pot să fie îndrumate în rețea doar către un singur partener la un moment dat. Pentru execuția regăsirilor și modificărilor, arhitectura suportă doar comunicări consecutive cu toate sistemele de calcul adresate în cursul regăsirii sau modificării dorite. Totuși baza de date nu este organizată astfel încît doar doi parteneri să poată comunica pentru execuția unei cereri de regăsire ori de modificare. Se impune existența posibilității de execuție consecutivă a comunicației între calculatorul-gazdă emițător al cererii și un număr de alte calculatoare-gazdă. În acest moment, nici un mecanism de transmitere nu este suportat de nivelul sesiune sau la un alt nivel inferior. Trebuie ca la nivelul SGBDD să fie implementat un software suplimentar, luîndu-se evident în considerație și overhead-ul adițional de operare a SGBDD-ului.

Pentru tehnologia actuală a rețelelor, cu comunicații cu o lungime de bandă relativ mică, orice comunicație necesară în cursul servirii unei cereri, va fi mult mai lentă decît comunicarea dintre procesele din același sistem de calcul. Se știe că timpul de răspuns pentru realizarea regăsirilor și modificărilor într-o aplicație interactivă de bază de date pe un sistem centralizat a ajuns la o valoare intolerabil de cerere. Cu atît mai mult va fi inacceptabilă întîrzierea pentru execuția consecutivă a comunicațiilor în interiorul rețelei de bază distribuită. Rezultă că aplicațiile care conduc la un grad ridicat a dispersiei datelor și a cererilor de regăsire/modificare vor fi în primul rînd limitate de canalul de comunicație. Doar acele aplicații de baze de date care implică un grad scăzut de dispersie pot fi executate pe rețelele actuale de comunicații publice.

Arhitectura sistemelor deschise a fost gîndită să suporte o gamă largă de aplicații în rețelele de calculatoare. Serviciile pentru conversia și transformarea formatelor și structurilor de date diferite sînt localizate la nivelul 6 (prezentare) al modelului arhitectural. Orice sistem de bază de date distribuit peste mai multe localizări sau orice sistem de bază de date distribuit, localizat în calculatoare-gazdă de diferite tipuri se presupune că utilizează serviciile furnizate la nivelurile 1 la 6. Toate serviciile furnizate de aceste niveluri vor fi suportate permanent de calculatoare-gazdă din sistemul deschis, chiar dacă un mare număr din ele nu vor fi necesare în gestiunea BDD. Overhead-ul care rezultă va determina atît o creștere a încărcării unităților centrale ale calculatoarelor-gazdă [D-11], cît și o creștere a volumului de comunicații. Aceasta, în schimb, va conduce la întîrzieri în execuția cererilor de regăsire/modificare, care nu vor fi tolerate de aplicațiile interactive de baze de date. Rezultă că aplicațiile cu un grad pronunțat de eterogenitate din punctul de vedere al bazelor de date și al sistemelor gazdă vor fi în primul rînd limitate de unitatea centrală de prelucrare. În rețelele publice vor putea fi implementate doar aplicațiile de baze de date care au grad scăzut de eterogenitate a bazelor de date și a calculatoarelor-gazdă.

În calitate de tendințe actuale în dezvoltările privind gestiunea BDD putem menționa:

— dacă datele nu sînt specifice unui utilizator și nu pot fi localizate în nodul „utilizatorului principal", ele vor fi în mod obișnuit plasate central, pentru a evita un grad de dispersie ridicat a cererilor de regăsire și de modificare. Orice sistem simplu de comunicații de date este în mod obișnuit prevăzut cu o bază de date centralizată pentru a suporta cererile de servicii emise de la localizările aflate la distanță;

— dacă sistemul de bază de date este eterogen în raport cu sistemul de bază de date locală sau cu sistemul de calculator gazdă, sistemul va fi proiectat ca un SGBDD dedicat pentru evitarea overhead-ului necesar. Doar aceste sisteme asigură performanțe acceptabile (gradul de dispersie în cererile de regăsire și modificare, strategiile de localizare a datelor, gestiunea directorilor, tehnici de control al accesului și integrității BDD), în pofida măririi volumului calculelor necesare conversiei și transformării datelor;

— dificultățile întîlnite la sistemele de BDD puternic eterogene au condus chiar la ideea rescrierii și redezvoltării acelor sisteme într-o manieră omogenă.

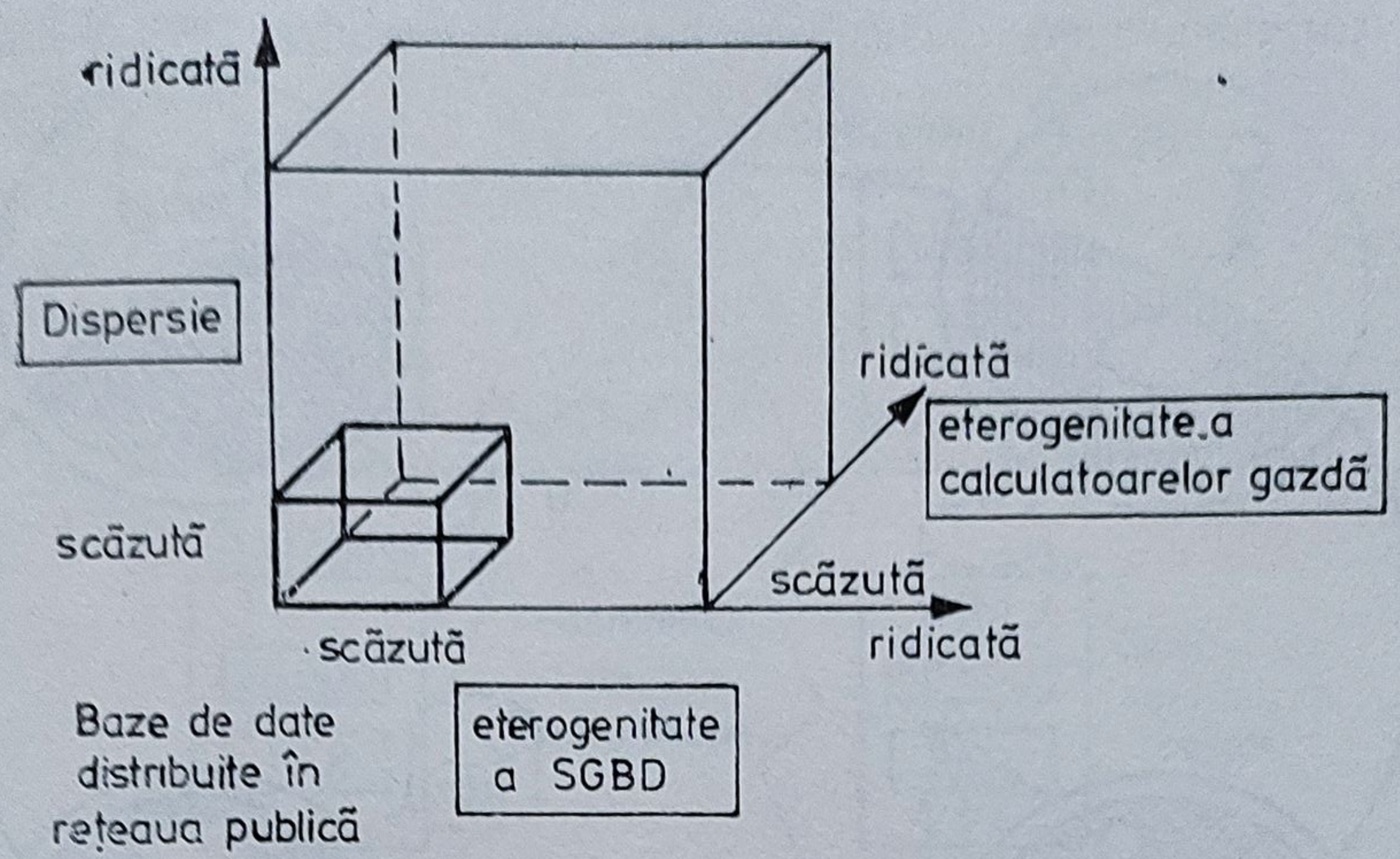


Fig. 6.15.

În concluzie, luînd în considerație realitățile și tendințele din gestiunea sistemelor cu date distribuită, în fig. 6.15 sînt recomandate o serie de caracteristici pentru BDD din rețelele publice.

6.3.3. VIRTUALIZAREA BAZELOR DE DATE ETEROGENE

Sistemele de baze de date localizate fizic în diferite noduri din rețea folosesc un hardware și un software foarte diversificate și permit accesul prin intermediul unei multitudini de limbaje de acces și cu diferite modele de date (subscheme) pentru utilizatori. Astfel, utilizatorii rețelei trebuie să ia în considerație această eterogenitate a sistemului de bază de date și caracteristica distribuției datelor respective.

Pentru rezolvarea problemei eterogenității și a distribuției datelor va fi introdus conceptul de virtualizare a SGBDD-urilor și utilizatorilor. O cale de structurare a sistemelor de bază de date distribuite printr-o abordare de jos în sus (bottom-up) este reprezentată de soluționarea eterogenității și a distribuției datelor. Conceptul de virtualizare este introdus ca o modalitate de rezolvare optimă a acestor două probleme. Tehnicile „virtualizării" sînt folosite pe larg în domeniul comunicațiilor între calculatoare. Putem menționa, de asemenea, unele activități inițiate de ISO și CCITT de a virtualiza terminale și alte resurse ale rețelelor de comunicație. În prezenta lucrare va fi folosit conceptul de virtualizare la baza de date distribuită, considerînd că înseși datele memorate în baza distribuită constituie o resursă a rețelei. Potrivit conceptului de virtualizare, un sistem de bază de date distribuită constă în utilizatori virtuali și executivi virtuali-numiți cu toții resurse virtuale (fig. 6.16).

Virtualizarea presupune următoarele două abordări:

• *virtualizare pentru eterogenitatea resurselor*. O resursă virtuală are acces la altă resursă virtuală, fără a fi conștientă de eterogenitatea ei, ca și cînd ar avea acces la o rețea cu resurse omogene. Astfel, atunci cînd utilizatorul

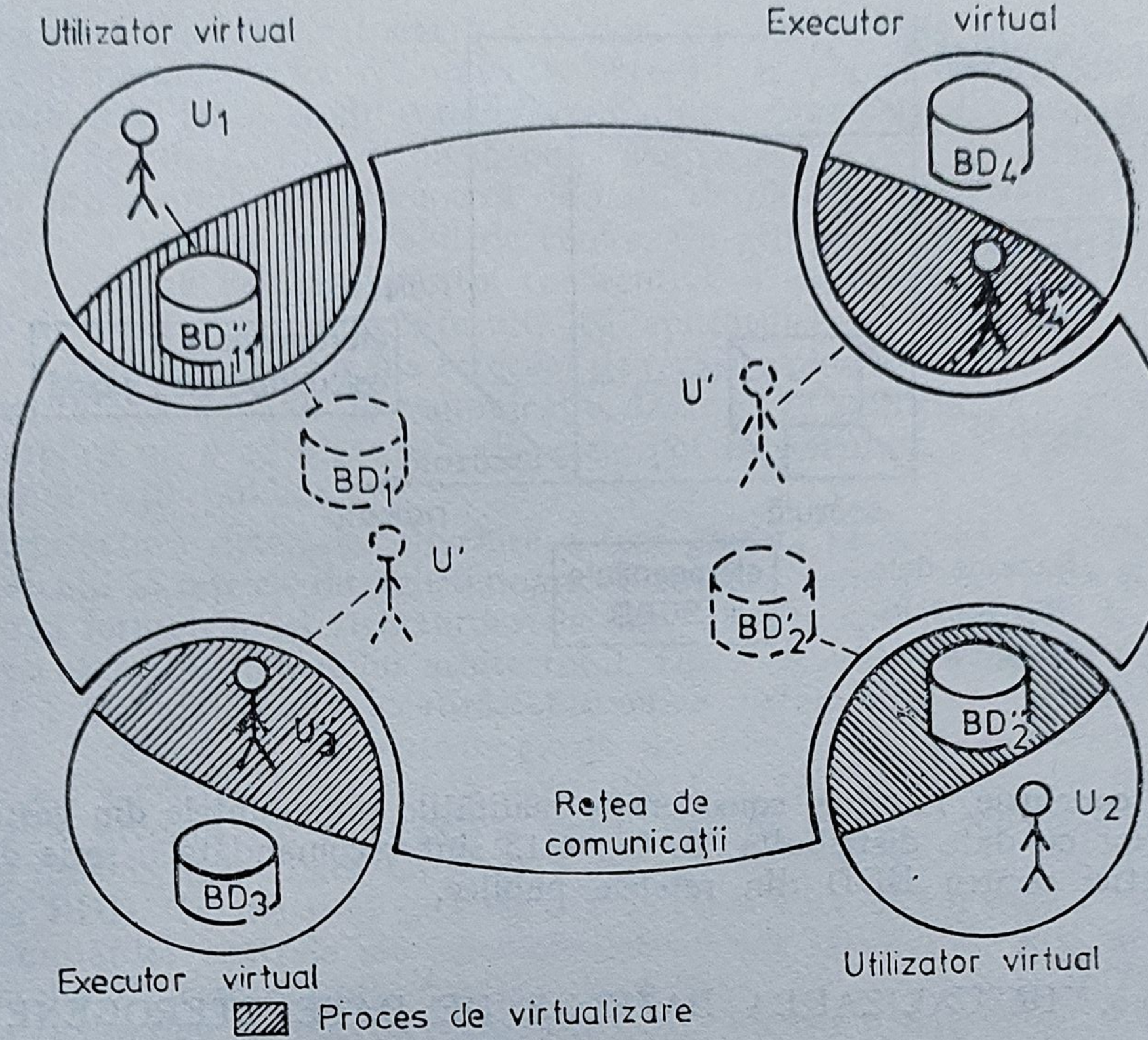


Fig. 6.16.

virtual comunică cu executorul virtual utilizatorul nu are nevoie să aibă cunoștință despre schema și limbajul de acces utilizate de executorul real. În mod dual, cînd un executor virtual comunica cu un utilizator virtual executorul nu trebuie să posede cunoștințele despre interfața folosită de utilizatorul real, ca de exemplu un limbaj de interogare de nivel înalt.

• *virtualizarea pentru distribuția datelor.* Utilizatorul virtual poate să vadă sistemul de bază de date distribuită, fără a cunoaște distribuția datelor, ca și cum el ar avea acces la un singur executor.

6.3.4. ARHITECTURA UNEI BAZE DE DATE DISTRIBUITE

Arhitectura unei rețele a sistemului de baze de date distribuită din punctul de vedere al tehnicilor de comunicație și al exploatării bazei de date este dată în figura 6.17. O rețea de bază de date distribuită constă din utilizatori virtuali și executori virtuali, care asigură, facilități de utilizatori virtuali, respectiv, facilități de executori virtuali, și o rețea ce furnizează facilități ale rețelei de comunicație. Cu toate că atribuirea fizică a resurselor virtuale nu este permanentă, în mod foarte simplu un utilizator virtual este alocat la un nod de utilizator, un executor virtual la un nod de execuție și o rețea la un subsistem de comunicație. Un nod poate să aibă atît facilități de utilizator virtual, cît și de executor virtual. Cînd o rețea are un centru de

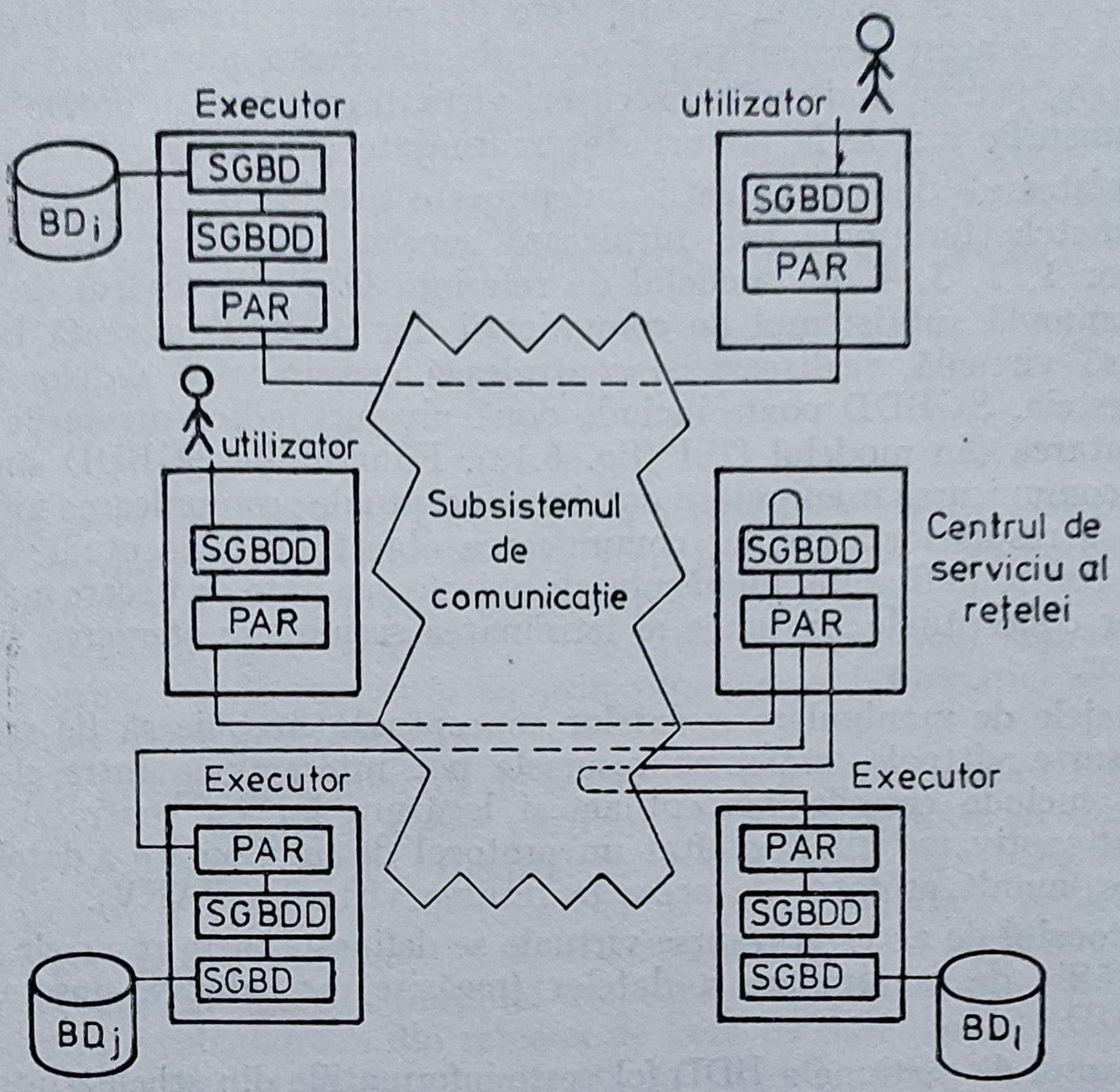


Fig. 6.17.

servicii ale rețelei similare celuia din figura 6.17 acesta posedă ambele facilități, fiind furnizată utilizatorilor o viziune integratoare asupra datelor fără ca el să aibă cunoștință asupra localizării acestor date. Astfel, la început este transmisă o cerere de la un nod utilizator, ce acționează ca un utilizator virtual, către un centru de serviciu al rețelei — în calitatea de executor virtual; apoi cererea este transmisă de la acest centru de serviciu — acum pe post de utilizator virtual — la un nod executor, ce se constituie în executor virtual.

Facilitatea de utilizator virtual constă în transformarea unei manipulări a datelor, ca de exemplu o interogare din limbajul orientat pe utilizator, într-o manipulare a datelor conceptuale folosind informații de eterogenitate precum schema externă locală și schema externă globală. Totodată, aceste facilități translatează obiectul conceptual, care constituie răspunsul interogării conceptuale, în structura de date de ieșire vizibilă utilizatorilor.

Facilitatea de executor virtual transformă manipularea datelor conceptuale în procedurile de acces orientate pe SGBD-ul local, folosind informația de eterogenitate, ca de pildă schema conceptuală locală și schema internă locală, pentru a avea acces la baza de date și de a transforma răspunsul ei în obiect conceptual.

Facilitățile de utilizator/executor virtual la nivelul centrului de serviciu al rețelei sînt oarecum diferite față de acelea ale utilizatorului și executorului virtual descrise mai înainte. O manipulare a datelor conceptuale

provenite de la utilizatorul virtual este descompusă folosind informația de interogare. La acest moment pot fi generate mai multe manipulări ale datelor conceptuale, pentru mai mulți executori virtuali care conțin datele solicitate, iar răspunsurile lor de la noduri sînt rearanjate și interclasate.

Subsistemul de comunicații îndeplinește sarcina transferului de date între punctele terminale sau furnizează servicii transport care corespund nivelurilor 1, 2, 3, 4 din modelul de referință OSI. Procesorul de acces la rețea, comandă subsistemul de comunicații, iar SGBDD execută facilitățile de resursă virtuală, realizează și controlează comunicația datelor și rezultatul acesteia. SGBDD poate include două niveluri, adică nivelurile sesiune și prezentarea din modelul OSI (fig. 6.18). Funcțiunile SGBDD sînt următoarele: comunicarea manipulării datelor conceptuale; comunicarea informației de stare a resurselor virtuale; comunicarea obiectului conceptual; controlul erorilor; controlul fluxului numărului de obiecte generate de fiecare manipulare a datelor conceptuale; inițierea și terminarea sesiunilor; ștergerea și oprirea proceselor.

Mesajele de manipulare a datelor conceptuale, trebuie să fie schimbate între resurse virtuale, astfel că resursele pot interacționa între ele. Interacțiunea include cererile conceptuale și legătura lor cu controlul asociat. Din acest motiv trebuie dezvoltat un protocol de manipulare a datelor standardizate, numit protocol de acces la resurse virtuale (PARV).

Protocolul de acces la resurse virtuale se definește între resursele virtuale și facilitățile de manipulare a datelor (regăsire, actualizare, inserare etc.) și SGBDD.

Sistemul de gestiune a BDD folosește informațiile din schema internă/externă locală, schema conceptuală locală, schema externă globală și informația de eterogenitate, pentru a pune în legătură obiectivele externe cu obiectivele

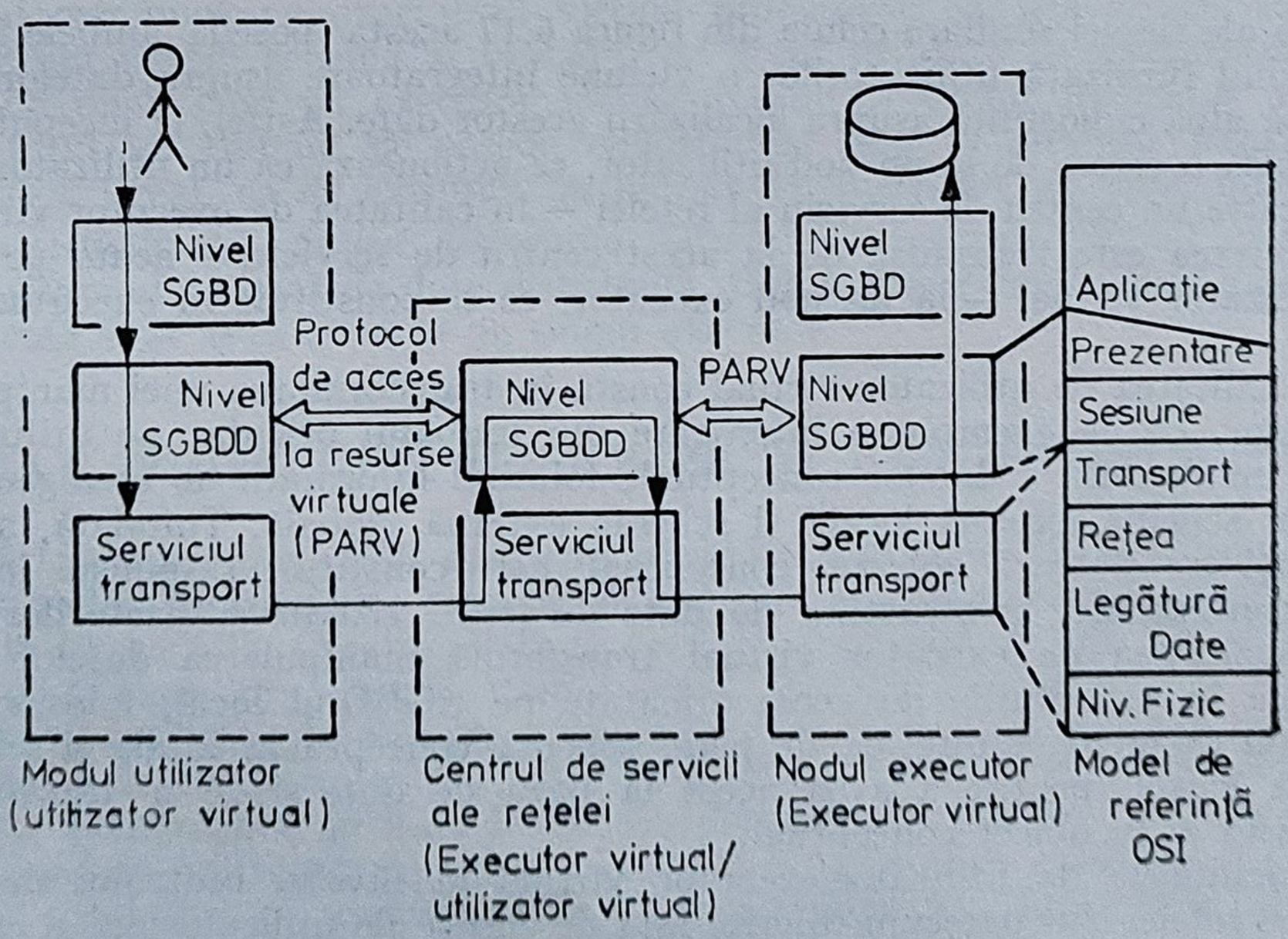


Fig. 6.18.

conceptuale. El implică astfel referiri la informația de integrare și realizează legături între obiectele conceptuale globale și obiectele conceptuale locale. PARV este folosit de SGBDD în cadrul nivelurilor de prezentare și sesiune. În principal, comenzile și răspunsurile legate de facilitatea de manipulare a datelor sînt definite similar cu protocoalele nivelului prezentare, iar cele relative la gestiunea rețelei de bază de date sau facilitățile de control al procesului sînt definite la fel ca și protocoalele nivelului sesiune.

6.3.5. SINCRONIZAREA TRANZACȚIILOR DE ACTUALIZARE

În realizarea unei aplicații care implică o BDD cea mai dificilă problemă constă în sincronizarea tranzacțiilor de actualizare. Este deci necesar de a imagina tehnici pentru controlul tranzacțiilor concurente, astfel încît să se asigure consistența bazei de date, în condițiile evitării unei supraîncărcări excesive a canalelor de comunicație, prin propagarea informației de control între nodurile bazei distribuite.

Cea mai la îndemînă metodă de sincronizare a actualizărilor distribuite constă în blocarea simplă a acelor porțiuni ale bazei de date în care citesc sau înscriu tranzacțiile active. Dealtfel acest mecanism este folosit în mod obișnuit pentru a actualiza sincronizările în bazele de date nedistribuite, convenționale. Într-o configurație de BDD, apare însă o întîrziere mare și uneori inacceptabilă datorită timpului de propagare a informației de blocare, către mai multe calculatoare din rețeaua de bază de date. Astfel un algoritm simplu de blocare distribuită, menționat în [K—4], necesită $5 \cdot n$ mesaje între calculatoare pentru a sincroniza o tranzacție între n noduri a unui sistem de BDD, precum urmează: n mesaje de solicitare a blocării, n confirmări de blocare, n mesaje pentru transmiterea actualizărilor, n confirmări de actualizare și n mesaje de eliberare a blocărilor. Dealtfel întîrzierea în executarea actualizării este mare, fiind compusă din întîrzierea maximă întîlnită în stabilirea blocărilor, plus întîrzierea maximă datorită realizării efective a actualizării. Întîrzierea de comunicație pentru mesaje într-o rețea de tip Arpanet ce folosește acest algoritm poate varia între 0,1 și 1 secundă, fiind cu două sau trei ordine de mărime mai mare față de întîrzierea tipică pentru stabilirea blocărilor într-un sistem centralizat. Efectul net al întîrzierii datorită blocării asupra timpului total de tranzacție depinde de detaliile algoritmului de prelucrare, reducînd, după o evaluare moderată, randamentul procesului de actualizare cu un ordin de mărime. Există puține sisteme de aplicații care pot să se adapteze unui asemenea grad de degradare a performanțelor, acesta fiind și motivul pentru care abordarea metodei de blocare simplă este inadecvată pentru o BDD de scop general. Pentru rezolvarea acestei probleme au fost propuse o varietate mare de soluții. Dintre ele menționăm:

- Soluții bazate direct pe conceptul clasic de blocare, prin care se caută să se găsească algoritmi mai eficienți pentru propagarea informației de blocare prin rețeaua de bază de date. Astfel s-a obținut un volum de comunicații doar de $1{,}5 \cdot n$, prin introducerea însă a unor întîrzieri mari de comunicație. Aceste soluții necesită totodată un volum mare de comunicații intercalcula-

toare pentru realizarea actualizărilor, nefiind performante în rețelele cu un număr mare de locații și un volum indicat de tranzacții;

• Abordări care caută să reducă costurile de sincronizare prin introducerea unui anumit nivel de control centralizat în sistemul distribuit. Aceste metode, numite ale „locului primar", necesită ca toate activitățile de actualizare pentru un fișier dat să fie realizate printr-o singură locație de bază de date, numit locul primar. Se obțin rezultate bune pentru aplicații simple de baze de date, în care este posibilă împărțirea efectivă a activității bazei de date în regiuni geografice. În aplicațiile mai complexe metoda nu poate evita totuși nevoia de blocare globală a bazei de date în condițiile unui cost ridicat de comunicație și ale unei întîrzieri mari;

• O altă direcție de acțiune în rezolvarea problemei sincronizării actualizărilor constă în încercarea de a evita blocarea globală ori de cîte ori este posibil. Metoda se bazează pe o analiză formală a căilor în care tranzacțiile pot să interfereze într-o BDD și a modalităților prin care poate fi evitată această interferență. Metoda este implementată la sistemul SDD-1 (Sistem de baze de date distribuite la Computer Corporation of America). Actualizarea se face prin intermediul mai multor „protocoale de sincronizare" ce oferă diferite niveluri de control al sincronizării, la un cost variat. Protocoalele de sincronizare sînt de fapt niște algoritmi care specifică ce trebuie făcut pentru a asigura o prelucrare corectă a unei tranzacții date. Cel mai eficient dintre protocoale nu specifică nici o sincronizare între calculatoare. O tranzacție care este prelucrată conform acestui protocol are nevoie doar de realizarea unei *blocări locale* pentru a asigura atomicitatea operațiilor ei de citire/scriere la fiecare loc individual. Fiecare din celelalte protocoale introduce un anumit grad de sincronizare nelocală, culminînd cu cel mai puternic protocol ce realizează un control mai puternic decît blocarea globală a bazei de date. Nu toate tranzacțiile pot fi executate sub cel mai eficient protocol. Operarea corectă a sistemului depinde în mod critic de selecția corespunzătoare de către sistemul de baze de date, a protocolului ce trebuie folosit în prelucrarea tranzacțiilor emise de utilizatori.

Metodologia de actualizare a sistemului SDD-1 permite o execuție a tranzacțiilor de actualizare mai rapid și la un cost mai scăzut față de utilizarea oricărei alte metode alternative.

6.3.6. CONSIDERAȚII PRIVIND PROIECTAREA BDD

Alocarea fișierelor reprezintă acțiunea de alocare a unor componente a bazelor de date către procesele din nodurile rețelei. Pentru rezolvarea cerințelor de alocare a fișierelor se aplică tehnicile clasice de programare matematică pentru diferite variante a următoarei probleme: (i) Se dă o descriere a cererilor declarate de servicii ale utilizatorilor (de exemplu, volumul regăsirilor și actualizărilor de la fiecare nod al rețelei) către fiecare fișier; (ii) Se dă o descriere a resurselor de care se dispune pentru a suporta această cerere, furnizînd informații despre topologia rețelei, capacitatea și costul legăturilor și capacitatea și costul exploatării nodurilor; (iii) Se cere să se determine o atribuire a fișierelor la nodurile rețelei care nu violează vreo constrîngere de capacitate și minimizează costurile totale. Unele variante ale acestei probleme care au fost minuțios explorate cuprind: considerații asupra cererilor

aleatorii și variabile în timp, considerații asupra problemei duale a constrîngerilor de cost și de determinare a capacităților, folosirea unor metode euristice de reducere a complexității computaționale pentru găsirea unei soluții accesibile.

Din nefericire problema alocării fișierelor într-o configurație de baze distribuită îmbracă aspecte mult mai complexe față de cele menționate mai înainte, putîndu-se identifica trei deficiențe specifice care limitează folosirea rezultatelor problemelor de mai sus în contextul BDD:

• Modelul de cerere a utilizatorului este considerat ca un set de cerințe de acces la un fișier dat, dintr-un nod dat. Acest model nu reflectă în mod adecvat cererea utilizatorului pentru accesul la o bază de date care implică mai mult decît un fișier. Amploarea costurilor de comunicație în această situație nu este descrisă într-un model care consideră doar comunicația între nodul utilizator și nodul fișier.

• Rezultatele actuale obținute pentru problema alocării fișierelor neglijează complet costurile de sincronizare în actualizarea datelor memorate redundant. Acest cost este foarte probabil că domină costul de actualizare și va varia corespunzător schemei de sincronizare alese, în funcție de mix-ul traficului de actualizare și într-o manieră complexă cu proiectarea băncii de date.

• Presupunerea că unitățile de alocare a datelor către nodurile rețelei trebuie să fie constituite din fișiere complete este inflexibilă. Există unele situații în care — permițînd partiționări orizontale sau verticale a fișierelor — se va reduce volumul de date la care se are acces și costul memoriei. Rezultă că în contextul unei BDD moderne în care coexistă tranzacții multifișier complexe rezultatele actuale ale cercetării nu furnizează instrumente adecvate pentru proiectarea bazelor de date.

Localizările bazei de date. Alegerea locurilor din rețea unde vor fi memorate datele din bază se realizează avînd ca obiectiv minimizarea volumului de trafic de comunicație. Trebuie să fie permisă legătura între localizările diferite ale bazei de date. Pentru a minimiza traficul între punctele de localizare se procedează la duplicarea unor date din bază. De exemplu, se pot duplica unele înregistrări ale listei de materii prime și materiale dintr-o aplicație cu o bază de date și înregistrările dintr-o bază de date tehnologică pentru urmărirea execuției utilajelor atît la proiectantul utilajelor, cît și în secția de fabricație propriu-zisă. Datele duplicate sînt mai ușor de manipulat față de punctatorii inter-localizării bazei de date. Apare însă o problemă nouă — asigurarea că toate copiile sînt actualizate deodată. Prin aceasta se previne actualizarea locală sau duplicarea înregistrărilor independent de celelalte localizări.

Regenerarea sistemului și a bazei de date. Mecanismul de regenerare (recuperare a unui sistem centralizat de bază de date implică blocarea datelor față de accesul proceselor active și înregistrarea schimbărilor efectuate în bază, împreună cu sincronizarea proceselor cu baza de date după producerea căderilor. Aceste mecanisme au fost descrise pe larg în literatura de specialitate. Distribuirea prelucrărilor creează două noi probleme. În primul rînd, într-un sistem de bază de date distribuite fiecare locație va realiza propria ei regenerare, folosind tehnicile sistemelor cu o singură bază de date, fiind însă în plus necesar să se sincronizeze prelucrările pentru toate localizările bazei de date. Aceasta se realizează mai întîi prin împărțirea task-ului de

prelucrat în unități independente sau „unități obligate", cu proprietatea că sistemul își ia răspunderea să prelucreze toate ori nici una din aceste unități. Astfel, dacă prelucrarea unei asemenea unități nu se poate face dintr-o anumită cauză, atunci se reia procesul de la începutul execuției activității respective și toate schimbările efectuate în baza de date sînt derulate înapoi.

Într-o bază de date distribuită sfîrșitul prelucrării unei unități, trebuie să fie făcut cunoscut tuturor localizărilor afectate de unitate. Aceasta implică o primă etapă, în care fiecare localizare a bazei este informată de terminarea prelucrării unității, cerîndu-i-se să „protejeze" procesul astfel încît atît terminarea, cît și rularea înapoi a schimbărilor să poată fi garantată. Doar atunci cînd toate localizările au confirmat starea „protejată" a procesului poate fi asigurată terminarea cu bine a prelucrării unității de bază. Doar în această etapă la fiecare localizare a datelor din bază se realizează schimbările obligatorii și se deblochează accesul la datele solicitate de execuția proceselor concurente.

Un alt eveniment care poate să se manifeste într-o bază de date distribuită, cu duplicarea datelor, este fragmentarea rețelei. Astfel, dacă rețeaua se desparte în două părți, fiecare jumătate își imaginează că cealaltă parte a fost distrusă și își continuă actualizarea propriei copii de date. Cînd cele două fragmente de rețea se reunesc va trebui să se realizeze reconcilierea celor două copii. Dacă s-a decis că trebuie să fie accesibili o copie „master", atunci numai o parte a rețelei va continua să lucreze. Copia „master" poate fi desemnată printr-un semn, astfel încît nu este în mod necesar limitată la o localizare particulară în rețea a datelor din bază.

Problema aceasta poate fi rezolvată satisfăcător prin sistemul de proiectare a bazei distribuite. Anumite tipuri de tranzacții sînt dependente de timp, în sensul că efectul lor asupra bazei de date depinde de contextul bazei de date și de aici de secvența în care apar aceste tranzacții. Alte tranzacții sînt independente de timp și pot fi aplicate bazei de date într-o secvență arbitrară. Este evident că dacă există probleme în comunicațiile în interiorul rețelei, schimbările independente de timp pot fi salvate, ele actualizează copiile printr-un program de reconciliere. Tranzacțiile dependente de timp vor fi aplicate doar atunci cînd toate copiile sînt sincronizate. Unele sisteme distribuite permit operarea în condițiile fragmentării rețelei, folosind reconcilierea, cel puțin pentru datele proprii ale sistemului de operare care supervizează lucrul SGBDD.

Performanțe. În sistemele convenționale de baze de date programele de aplicație și SGBD pot avea acces la același grup de date din memorie. Aceasta este echivalent cu a poseda o mașină-bază de date, la distanță, cu o lărgime de bandă infinită. Este deci evident că interfața pentru programele care comunică cu SGBD nu este optimă la sistemele distribuite. La un sistem distribuit care este optim, este necesar să se minimizeze traficul de date între localizări, fiind întîlnite și aici mașinile bază de date (back-end). Sînt două modalități de realizare a acestor baze, modalități care nu se exclud mutual. Prima modalitate constă în a permite mașinii bază de date să selecteze datele necesitate de programul de aplicație cît de precis este posibil. A doua cale asigură programului de aplicație posibilitatea de a executa taskuri logice chiar lîngă datele implicate. Aceasta se face reunind comenzile

separate de manipulare a datelor și trimițînd apoi un lanț al acestora spre a fi executat, înainte de a necesita un răspuns.

Administrare și control. Sistemele de aplicații ale bazelor de date distribuite combină probleme ale administrării rețelei și ale bazei de date necesitînd astfel un grad ridicat de asistență din partea sistemului de operare însăși. Pentru sistemele distribuite, operațiile de control efectuate de operator trebuie să fie de nivel înalt și consultative. Funcțiuni precum regenerarea datelor, sincronizarea sistemului după căderi și restaurarea sistemului trebuie să fie în întregime asigurate contra greșelilor de manipulare. Unul din beneficiile calculului distribuit constă în asigurarea unui control mai mare al utilizatorilor individuali asupra prelucrărilor lor efectuate local.

Pentru aceasta trebuie identificate care aspecte ale sistemelor necesită definiții specifice, reglementări și control și care pot fi variate pentru a se potrivi cerințelor locale. Cel mai important aspect al controlului rezidă în definirea și semnificația datelor cu care se prelucrează aplicațiile asupra bazei de date distribuite.

6.3.7. EXEMPLE DE SISTEME DE GESTIUNE A BAZELOR DE DATE DISTRIBUITE

Multitudinea și pluralitatea demersurilor pentru conceperea de sisteme de bază de date distribuite au condus la realizarea practică a unor sisteme de gestiune specifice, din care vom prezenta succint în continuare caracteristicile unora mai reprezentative.

• *Sistemul de gestiune SIRIUS-DELTA.* Este un SGBDD implementat pe o rețea locală de minicalculatoare, de tipul REALITÉ 2000, specializate în gestiunea bazelor de date. Acest sistem oferă utilizatorului instrumentele necesare pentru a descrie și manipula într-un singur ansamblu logic datele situate pe mai multe minicalculatoare. El gestionează tipuri diverse de repartizări și duplicări ale datelor, permite accesul concurent al datelor din bază și menține integritatea în caz de pană a uneia sau mai multor componente.

Arhitectura sistemului SIRIUS-DELTA comportă mai multe straturi, corespunzînd la patru niveluri funcționale: funcția de gestiune clasică a bazei de date (interpretarea cererilor, acces la datele locale); funcția de gestiune specifică contextului distribuit (descompunerea cererilor în funcție de localizare); funcția de control distribuit (coerența și toleranță la erori); funcția de execuție distribuită (activarea proceselor distribuite, transferul datelor). Aplicațiile acestei baze de date distribuite sînt mai ales în domeniul telematicii (biuroticii), sistemul de gestiune putînd fi folosit și pe rețele de comunicație mai elaborate (rețele locale de tip Ethernet sau rețeaua publică Transpac).

• *SGBDD pentru sistemul INGRES.* Ingres este un SGBD relațional rulînd sub sistemul UNIX, versiunea 7. S-a proiectat și implementat o versiune distribuită a sistemului relațional Ingres într-o configurație omogenă, ceea ce a impus ca toate calculatoarele să ruleze sub sistemul de operare Unix. S-au făcute două ipoteze.

a. Mai întîi trebuie să li se permită proceselor să ramifice (creeze) alte procese pe calculatoare la distanță. Ramificarea este calea prin care sînt create procese cooperative sub S.O. Unix. Interogările distribuite adresate sistemului Ingres, adică interogările asupra unor relații localizate în cel puțin două noduri diferite ale bazei de date, vor determina sistemul Ingres original să cheme toate sistemele Ingres de pe toate calculatoarele implicate de interogare. Sistemul original Ingres — acela care servește pe utilizator — este numit Ingres Master, iar sistemele chemate de acesta sînt numite Ingres Slave.

b. În al doilea rînd, s-a presupus că sistemele Ingres comunică prin canale (pipe), similar cu modul de comunicație al proceselor obișnuite. Deoarece Ingres e un SGBD relațional, datele sînt memorate sub formă de relații. Relația este unitatea de distribuție, ea fiind localizată doar pe un nod, neținîndu-se astfel nici-o copie. Pe fiecare sistem sînt menținute cataloage sub forma unor relații sistem speciale.

Informația chemată poate fi de forma: nume de relație; informație de analiză sintactică (domeniu, formate de nume etc.); informație de performanță (număr de tupluri); informația de consistență (protecție, integritate, constrîngeri). Pentru a se putea prelucra interogări distribuite, trebuie ca primele trei informații să fie prezente în fiecare mașină, iar informațiile de consistență, doar local.

• *Sistemul de gestiune POLIPHÈME.* Acest sistem de gestiune a BDD este implementat pe rețeaua CYCLADES ce conectează un set de calculatoare IRIS 80, sub S.O. SIRIS 8, folosind sistemul relațional URANUS. El prezintă următoarele caracteristici principale: distribuția datelor relaționale, descompunerea interogărilor, execuție și arhitectură distribuită.

Sistemul POLYPHÈME a fost dezvoltat ca o rețea de *mașini relaționale* (MR) abstracte, fiecare mașină fiind bazată pe trei componente principale:

— spațiul NAME, constînd din descrierea relațiilor și atributelor;

— spațiul OBIECT, alcătuit din solicitările relațiilor descrise în spațiul „nume";

— mecanismul MAPPING, care realizează corespondența între spațiile „obiect" și „nume".

Fiecare MR furnizează utilizatorului ei un limbaj de definire a datelor bazat pe algebra relațională. Există două tipuri de mașini relaționale:

— *mașină locală,* în care cele trei componente de mai sus sînt implementate pe același calculator real. În sistemul POLYPHÈME, acest tip de mașină este construit pe fiecare bază de date care cooperează, în scopul realizării unei comportări omogene;

— *mașină globală,* la care spațiul de „nume" este implementat pe un calculator, în timp ce spațiul „obiect" poate fi distribuit la o structură cu mai multe calculatoare, fiecare conținînd o mașină locală. Mecanismul de corespondență determină localizarea schimbărilor relațiilor. Prin intermediul unei mașini globale, utilizatorul sistemului POLYPHÈME poate defini și manipula o BDD (fig. 6.19).

Baza de date distribuită este definită ca un set de relații, relația fiind unitatea de distribuție ce este memorată într-o locație ce nu este cunoscută utilizatorului. Utilizatorul supune unei mașini globale cererile elementare (interogări, actualizări) sau o secvență de cereri ce se constituie într-o tranzacție. Sistemul analizează fiecare cerere, o transformă într-o structură in-

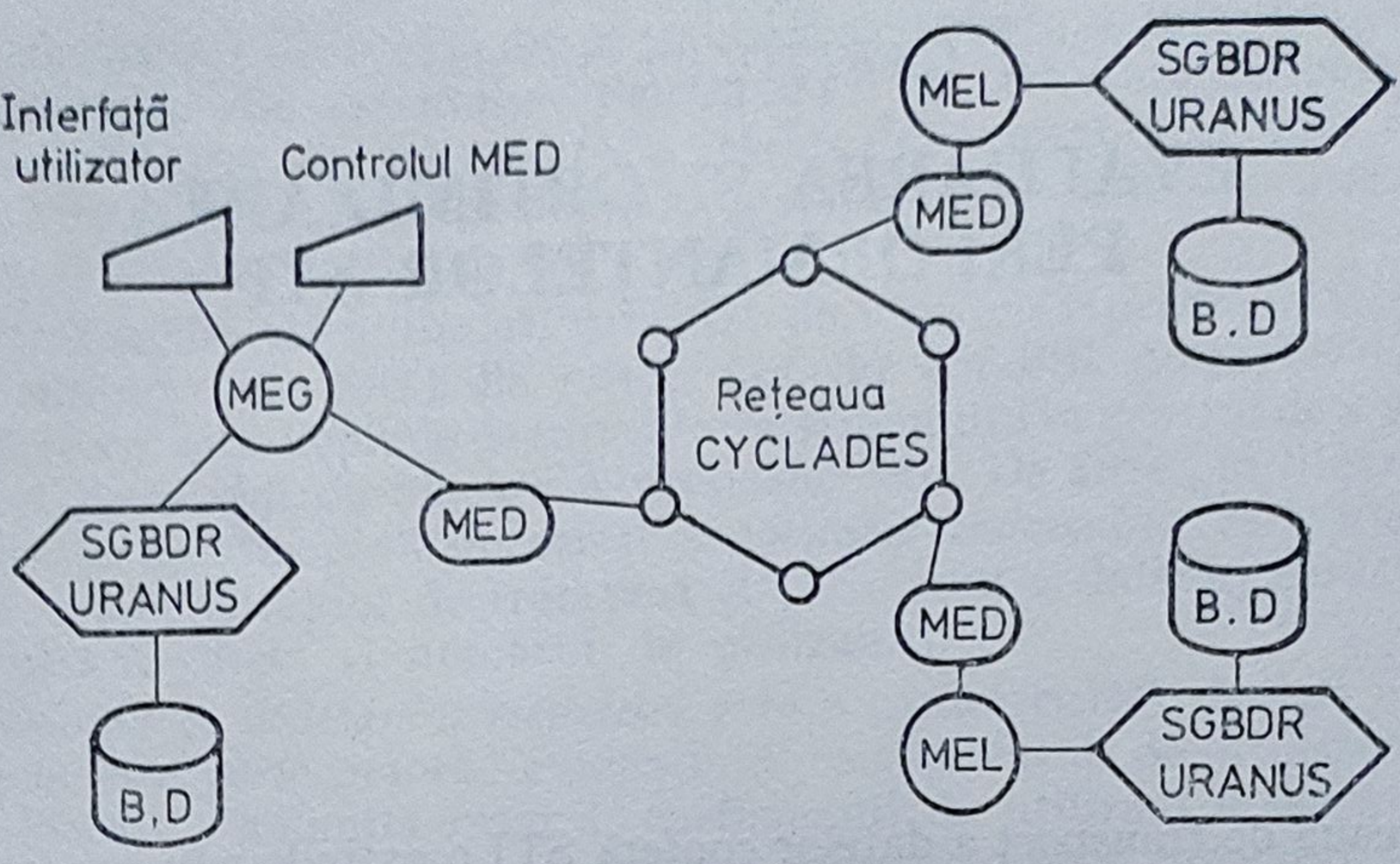


Fig. 6.19.

ternă de tip arbore, asupra căruia acționează algoritmul de descompunere. Rezultatul descompunerii este, de asemenea, un arbore la care fiecare nod (operator, relație) este atribuit la o localizare dată. Acum este posibil de a determina o serie de subarbori care privesc doar o singură mașină locală.

Implementarea mașinii locale și globale este legată de trei componente software specifice: SGBDR Uranus, monitorul de execuție distribuită și monitoarele de execuție locală și globală.

• *Sisteme de gestiune a BDD și comunicațiile prin satelit în cadrul proiectului NADIR.* După prima generație de SGBDD, din care fac parte și cele trei sisteme elaborate mai sus, [B-1, B-2, R-2] au apărut noi posibilități tehnologice (rețele locale de debit ridicat, sateliți de comunicație etc.). Tehnicile utilizate de BDD, ca de pildă prelucrarea interogărilor, algoritmi de control al concurenței și toleranța la defecte, sînt adaptate rețelelor terestre. Sateliții de comunicație existenți oferă un nou cadru pentru BDD, tehnicile folosite de SGBDD trebuind să fie adaptate caracteristicilor rețelelor prin satelit: o mare cuprindere geografică, debit ridicat, difuzarea informației și o întîrziere lungă de propagare. Proiectul de BDD-NADIR este situat în această perspectivă și utilizează serviciile sistemului francez de telecomunicații spațiale TELECOM 1. Acest sistem de telecomunicații prezintă o serie de caracteristici deosebite, precum: întîrzieri de propagare lungi (300 ms), debite ridicate de transfer (2 Mbiți/s), comutarea circuitelor (interfața *X*.21), circuite multipunct sau de radiodifuzare, o acoperire geografică foarte mare, procent de erori variabil și puțin cunoscut. Principalele probleme rezolvate de SGBDD-NADIR sînt:

— execuția optimală a cererilor care necesită accese la gestiuni diferite a BDD, localizate pe diferite calculatoare;

— algoritmi de control al concurenței, care protejează baza contra inconsistenței, în afară de actualizările concurente;

— toleranță la căderile din BDD.

Aceste sisteme prin sateliți prefigurează rețelele terestre ale viitorului, așa-numitele ISDN-uri, adică rețele numerice de date cu servicii integrate.

CAPITOLUL 7

EVALUAREA ŞI OPTIMIZAREA PERFORMANȚELOR STI

Rețeaua de transport a datelor conferă STI o extindere în spațiu, precum și unele caracteristici de performanță specifice — criterii ce definesc calitatea sistemului, a serviciilor oferite de către acesta. Pentru specialist, performanțele constituie parametri tehnici de proiectare, la care va trebui să răspundă sistemul și care, în final, vor dicta adoptarea unei anumite soluții dintre variantele propuse. Soluția însă trebuie să fie acceptabilă și din punctul de vedere al efortului pe care îl implică pentru realizarea ei. Prin urmare, performanțele sistemului vor fi corelate cu costul, proces prin care se va obține varianta optimă de proiectare. Performanțele trebuie să răspundă și unor cerințe de stabilitate în timp, refacere după incidente și chiar de îmbunătățire.

Odată cu definirea parametrilor de performanță este necesar să se elaboreze și metode de cuantificare a lor, stabilindu-se condițiile în care ei urmează să fie atinși.

7.1. PERFORMANȚELE STI

La caracterizarea STI concură multe criterii, fapt pentru care s-a propus împărțirea lor în două clase [M-1]: o primă clasă reunește performanțele ce caracterizează capacitatea de transport și prelucrare a sistemului, iar cealaltă — pe cele referitoare la siguranța sa în funcționare.

Principalele caracteristici de performanță ce aparțin primei clase sînt: timpul de răspuns; numărul mediu de utilizatori; coeficientul de utilizare a resurselor; stabilitatea; extensibilitatea; securitatea informației; informația stocată în baza de date; facilitățile oferite utilizatorului pentru prelucrarea datelor și afișarea informației.

Costul STI, deși nu apare ca performanță distinctă, este un parametru de sinteză, care se ia în calcul în analizele de eficiență.

Timpul de răspuns este durata măsurată între momentul emisiei unui mesaj într-un punct al sistemului și primirea răspunsului. Acest interval de timp se datorează propagărilor directe și inverse ale mesajului, precum și prelucrărilor la care acesta este supus în procesorul destinație. Un parametru ce caracterizează în exclusivitate viteza de transport a informației

în rețea este timpul de propagare a unui mesaj standard între cele mai „îndepărtate" puncte ale sistemului. Pentru STI cu comutare de mesaje (pachete), durata medie de traversare a rețelei de transport se mai numește și *timp de tranzit*, parametru ce depinde de capacitatea liniilor și a bufferelor, de tehnica de rutare utilizată etc.

Numărul mediu de utilizatori este dat de mulțimea persoanelor care pot beneficia simultan de serviciile sistemului, în condiții normale de lucru. Această performanță depinde de capacitatea sistemului în general de a satisface apelurile și de modul de alocare a resurselor etc. De altfel, în literatura de specialitate se menționează uneori și *capacitatea sistemului*, ca performanță de sinteză a posibilităților de transport și prelucrare a datelor, caracteristică ce influențează direct și numărul de utilizatori.

Coeficientul de utilizare a resurselor este o caracteristică deseori invocată în analizele cost-performanță, evidențiind randamentul de utilizare a sistemului. Pentru o anumită resursă, coeficientul de utilizare măsurat pe durata unei perioade de timp este dat de raportul dintre capacitatea medie efectiv utilizată și capacitatea maximă disponibilă.

Stabilitatea constă în capacitatea sistemului de a satisface apelurile utilizatorilor chiar și în momentele de vîrf ale traficului. Această performanță depinde de modul în care sistemul a fost dimensionat; un sistem supradimensionat va fi stabil chiar și în momentele de vîrf, iar timpul de răspuns nu va fi afectat. Dimpotrivă, în cazul unei subdimensionări timpul de răspuns poate fi considerabil alterat în momentele de trafic intens. Deoarece mărirea capacității sistemului implică creșterea costului, sarcina proiectantului este de a găsi soluția optimă, care să asigure o stabilitate bună la un preț acceptabil.

Extensibilitatea unui sistem constă în posibilitatea de a fi modificat în scopul măririi numărului de utilizatori sau al diversificării serviciilor pe care le oferă, fără a se altera parametri funcționali de bază. Extensiile nu trebuie să afecteze esențial arhitectura sistemului și nici să cauzeze discontinuități în funcționare.

Securitatea informației presupune capacitatea sistemului de a asigura atît protecția, cît și integritatea datelor vehiculate. Protecția se face prin reglementarea accesului la resurse pentru diferite categorii de utilizatori, pe cînd integritatea informației rezidă în capacitatea sistemului de a nu altera datele vehiculate.

Informația stocată în baza de date, modul de organizare și acces a acesteia, alături de facilitățile oferite de sistem pentru prelucrarea datelor constituie performanțe de natură software. De asemenea, pentru utilizator prezintă importanță modul de introducere a informației în sistem, precum și de afișare a datelor, cantitatea de informație afișată la un moment dat și timpul de expunere, posibilitatea asigurării unei copii a informației afișate etc. Între performanțele enumerate există o strînsă interdependență. De exemplu, timpul de răspuns este în dependență directă cu capacitatea sistemului, pe cînd stabilitatea se află în relație inversă cu coeficientul de încercare a resurselor; cu cît componentele sistemului vor funcționa mai aproape de capacitatea maximă, cu atît sistemul, în ansamblu, va fi mai instabil. Această contradicție rezidă și din faptul că unele performanțe, cum este timpul

de răspuns, sînt avute în vedere în primul rînd de către utilizator, dorindu-le a fi cît mai bune, iar altele, precum încărcarea resurselor, care implică probleme legate de eficiență și cost, sînt analizate în faza de proiectare.

7.2. ANALIZA DISPONIBILITĂȚII STI

Utilizatorul unui STI este deopotrivă interesat de performanțele sistemului definite anterior, cît și de menținerea lor în timp, de încrederea pe care o poate avea în buna funcționare a sistemului, încredere ce se bazează pe conceptul de *disponibilitate*.

Sistemele teleinformatice, îndeosebi cele mari, sînt incluse în categoria sistemelor de mare răspundere funcțională, pentru care asigurarea unei fiabilități ridicate devine o sarcină importantă de proiectare. Întrucît STI reprezintă o întrepătrundere a mijloacelor de comunicație cu cele de calcul, a căror funcționalitate este asigurată atît de componente hardware cît și software, analiza fiabilității lor este complexă.

7.2.1. INDICATORI DE FIABILITATE

În evaluarea disponibilității unui STI intervin atît indicatori de fiabilitate generali, valabili pentru orice sistem, cît și indicatori specifici sistemelor teleinformatice.

- **Principalii indicatori de fiabilitate pentru un sistem fără restabilire**

Funcția de repartiție a duratei de funcționare, notată $F(t)$, este probabilitatea ca sistemul, pus în funcțiune la momentul inițial 0, să se defecteze pînă la momentul de timp t. Notînd cu T timpul de bună funcționare a sistemului pînă la defectare, atunci:

$$F(t) = P(T < t). \tag{7.1}$$

Funcția de fiabilitate, notată $R(t)$, este probabilitatea ca sistemul să funcționeze fără defecte, în condițiile date, pînă la momentul t:

$$R(t) = P(T \geqslant t) = 1 - F(t). \tag{7.2}$$

Cunoscîndu-se fiabilitatea sistemului $R(t)$, se poate calcula valoarea medie pentru timpul de bună funcționare pînă la prima defectare, adică *media timpului de bună funcționare (MTBF)*:

$$MTBF = \int_0^\infty R(t)\, dt \tag{7.3}$$

Notînd $\rho(t + dt)$, probabilitatea ca sistemul să se defecteze în intervalul $(t, t + dt)$, în ipoteza că a funcționat pînă la momentul t, atunci conform formulei de calcul al probabilității unui eveniment condiționat rezultă:

$$\rho(t + dt) = \frac{P(t \leqslant T < t + dt)}{P(T \geqslant t)} = \frac{R(t) - R(t + dt)}{R(t)};$$

iar $$\lim_{dt\to 0}\frac{\rho(t+dt)}{dt}=-\frac{1}{R(t)}\lim_{dt\to 0}\frac{R(t+dt)-R(t)}{dt}=-\frac{dR(t)}{R(t)}.$$

Relația obținută, în care $dR(t)$ este derivata în raport cu t a fiabilității $R(t)$, se numește *intensitatea defectării instantanee* la momentul t și se notează cu $\lambda(t)$. Prin urmare:

$$\lambda(t)=-\frac{dR(t)}{R(t)}. \tag{7.4}$$

Din relația 7.4 se deduce că:

$$R(t)=exp\left(-\int_0^t\lambda(x)\,dx\right). \tag{7.5}$$

Practica a demonstrat că apariția defecțiunilor la un produs este descrisă de anumite funcții analitice tipice, dintre care cea mai frecvent utilizată este legea exponențială caracterizată de o intensitate a defecțiunilor constantă, caz în care:

$$R(t)=exp\,(-\lambda t),\quad t\geqslant 0; \tag{7.6}$$

$$MTBF=\frac{1}{\lambda}. \tag{7.7}$$

Această lege descrie apariția defecțiunilor întîmplătoare pentru echipamentele electronice în perioada de stabilitate în funcționare.

Întrucît un sistem este format din mai multe componente, fiecare componentă i fiind caracterizată de o fiabilitate $R_i(t)$, și dacă starea de funcționare a sistemului presupune funcționarea tuturor componentelor sale, atunci sistemul din punctul de vedere al fiabilității este un *sistem serie* (fig. 7.1)

Funcția de fiabilitate a întregului sistem va fi în acest caz:

$$R(t)=\prod_{i=1}^{n}R_i(t)=exp\left(-\sum_{i=1}^{n}\int_0^t\lambda_i(x)\,dx\right)=exp\left(-\int_0^t\sum_{i=1}^{n}\lambda_i(x)\,dx\right), \tag{7.8}$$

iar intensitatea defecțiunilor:

$$\lambda(t)=\sum_{i=1}^{n}\lambda_i(t). \tag{7.9}$$

Prin urmare, pentru ca fiabilitatea întregului sistem să fie ridicată este necesar ca toate componentele sale să aibă un coeficient de fiabilitate mare. Dar cum în cazul sistemelor tehnice mari o asemenea cerință este greu de realizat, se recurge la *tehnica redondării* componentelor mai puțin fiabile, realizîndu-se ansambluri *paralele*. Un asemenea ansamblu redondant cu m componente este prezentat în figura 7.2. Dacă funcționalitatea unui asemenea sistem este asigurată de funcționarea cel puțin a unui element, atunci funcția sa de fiabilitate, $R(t)$, va fi:

$$R(t)=1-\prod_{i=1}^{m}(1-R_i(t)). \tag{7.10}$$

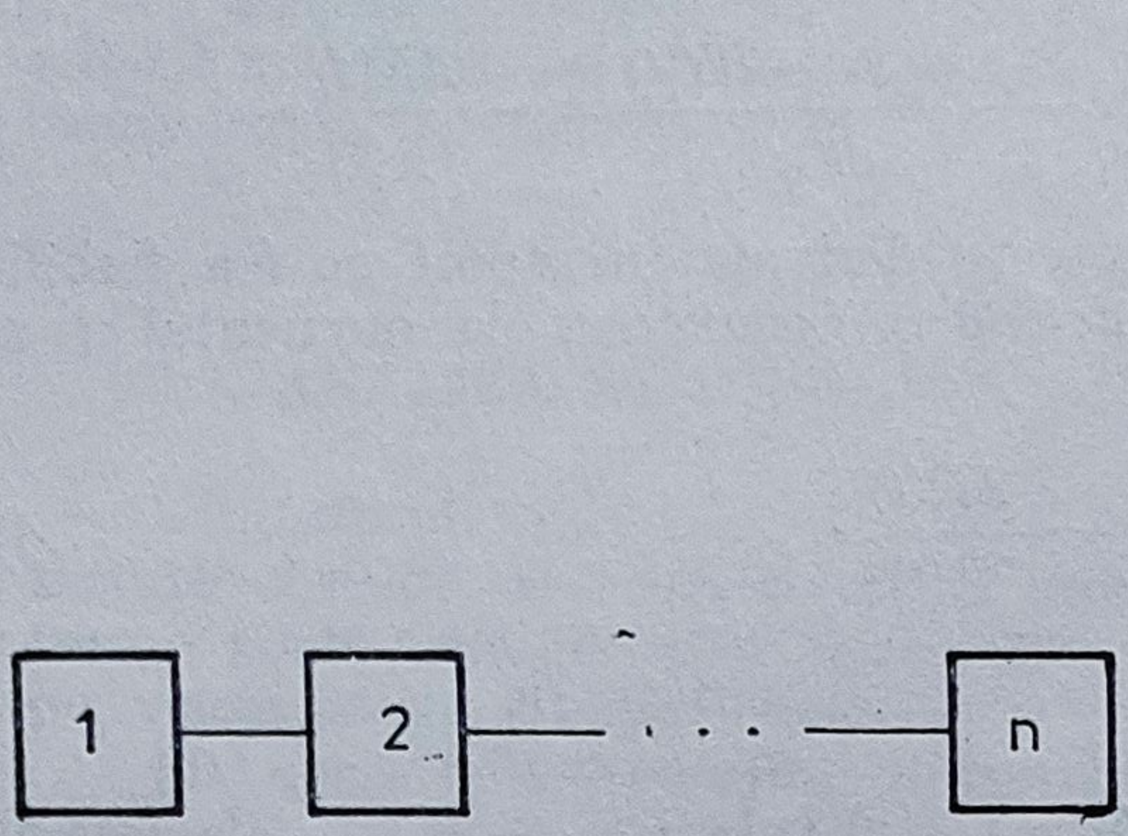


Fig. 7.1. Sistem serie cu n componente

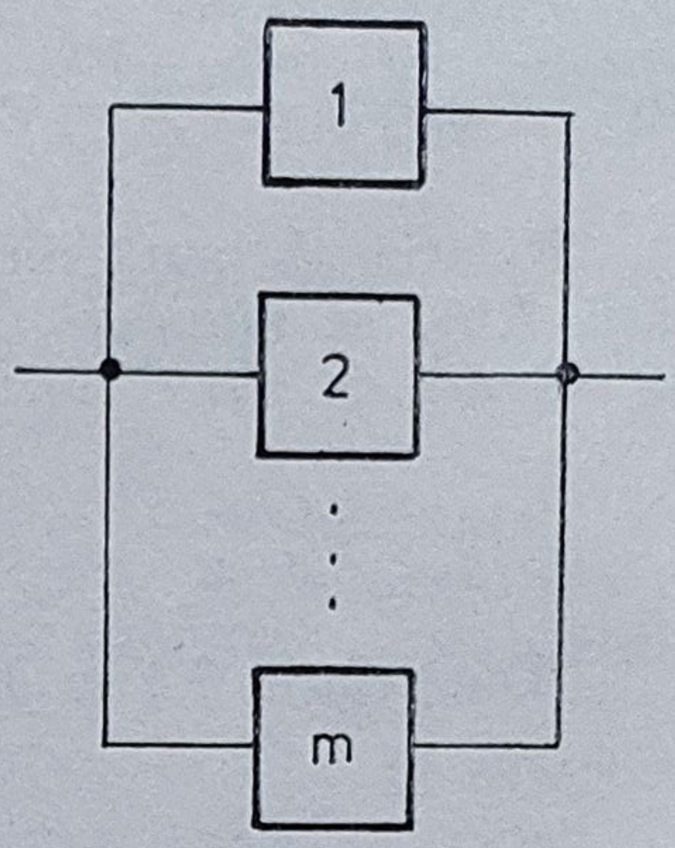


Fig. 7.2. Sistem paralel cu m componente

De exemplu, presupunînd $m = 2$ și $R_1 = R_2 = 0{,}9$ se obține pentru sistem următoarea valoare a fiabilității în regim staționar:

$$R = 1 - (1 - R_1)(1 - R_2) = 0{,}99.$$

- **Fiabilitatea unui sistem redondat reparabil**

O bună funcționalitate pentru un sistem se poate asigura atît printr-o valoare ridicată a fiabilității $R(t)$, cît și prin prevederea unor măsuri operative de identificare și remediere a deranjamentelor. Probabilitatea ca un sistem reparabil să fie în funcțiune la un moment dat t, avîndu-se în vedere și posibilitatea restabilirii lui, este *disponibilitatea* sistemului, $D(t)$, dată de relația:

$$D(t) = R(t) + \int_0^t R(t - x)\, r(x)\, dx. \tag{7.11}$$

Conform relației 7.11, pentru ca sistemul să funcționeze la timpul t este necesar ca acesta ori să nu se fi defectat pînă la acel moment, sau — în caz de defectare — să fi fost reparat. Practic, acest indicator măsurat pe intervalul de timp $(0, t]$ este dat de raportul dintre suma perioadelor în care sistemul a fost utilizabil și timpul total t. Pentru valori suficient de mari ale lui t, disponibilitatea $D(t)$ tinde către o valoare staționară D:

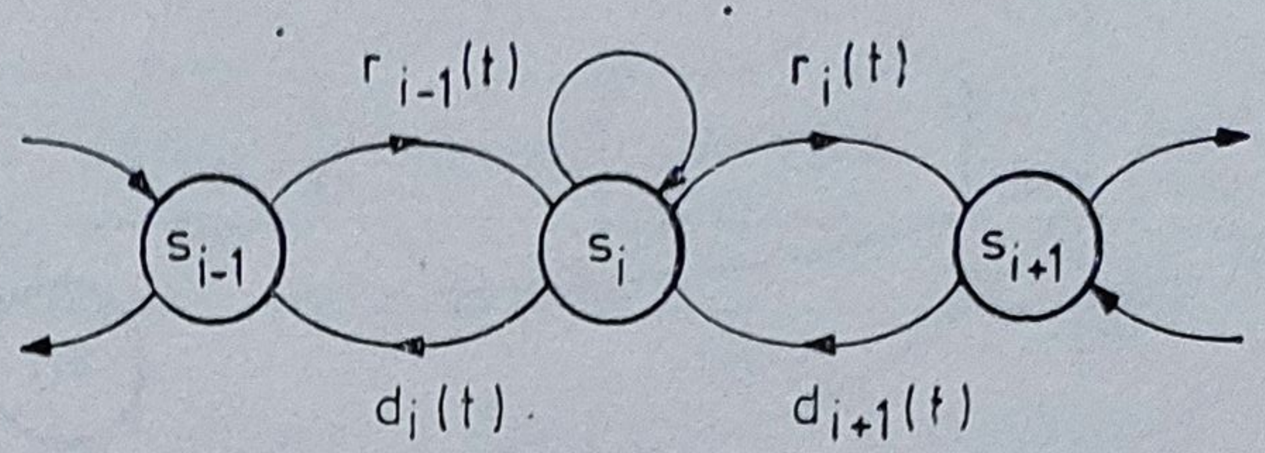
$$\lim_{t \to \infty} D(t) = D. \tag{7.12}$$

Pentru caracterizarea nefuncționalității unui sistem se folosește coeficientul de *indisponibilitate* I, care reprezintă ponderea timpului total de nefuncționare, astfel că:

$$I = 1 - D. \tag{7.13}$$

Considerînd un sistem redondant de ordinul n cu redondanță activă totală, atunci acesta va fi defect numai în cazul nefuncționării tuturor componentelor sale. Prin urmare, sistemul se va afla la un moment dat t într-una din cele $n+1$ stări posibile, starea s_i fiind caracterizată de funcționarea a n elemente,

Fig. 7.3. Graful tranzițiilor stărilor unui sistem redondant



restul elementelor fiind defecte. Notînd cu: $s_0, s_1, ..., s_n$ stările sistemului; $d_i(t)$ — densitatea probabilității de defectare a unei componente a sistemului aflat în starea s_i, iar $r_i(t)$ — densitatea probabilității de restabilire a unei componente defecte, atunci conform grafului tranzițiilor stărilor (fig. 7.3), probabilitatea ca sistemul să se afle la momentul $t + dt$ în starea s_i — notată $P_i(t + dt)$ — , considerînd că la timpul t era într-una dintre stările s_{i-1}, s_i sau s_{i+1}, este dată de relația:

$$P_i(t + dt) = P_{i-1}(t)r_{i-1}(t)dt + P_i(t)\,[1 - d_i(t)dt]\,[1 - r_i(t)dt] + P_{i+1}(t)d_{i+1}(t)_i. \tag{7.14}$$

Din relația 7.14 se obține:

$$\lim_{dt\to 0}\frac{P_i(t + dt) - P_i(t)}{dt} = \frac{dP_i(t)}{dt} = P_{i-1}r_{i-1}(t) - P_i(t)\,[d_i(t) + r_i(t)] + P_{i+1}(t)d_{i+1}(t). \tag{7.15}$$

Probabilitățile $P_i(t)$ satisfac condiția $\sum_{i=0}^{n} P_i(t) = 1$.

Relațiile 7.15 formează un sistem de ecuații diferențiale cu condiția inițială $P_n(0) = 1$, în ipoteza că la punerea în funcțiune a sistemului nici o componentă nu era defectă.

În cazul în care duratele de funcționare și de restabilire ale elementelor sistemului sînt repartizate după legi exponențiale de parametri λ, respectiv μ, atunci 7.15 devine:

$$\frac{dP_i(t)}{dt} = \mu_{i-1}P_{i-1}(t) - (\lambda_i + \mu_i)\,P_i(t) + \lambda_{i+1}P_{i+1}(t), \quad i = 1, ..., n - 1. \tag{7.16}$$

Pentru un ansamblu cu o singură componentă se demonstrează că soluția sistemului de ecuații 7.16 este:

$$P_0(t) = \frac{\lambda}{\mu + \lambda} - \frac{\lambda}{\mu + \lambda}\,exp\,(-(\lambda + \mu)\,t); \tag{7.17}$$

$$P_1(t) = \frac{\mu}{\mu + \lambda} + \frac{\lambda}{\mu + \lambda}\,exp\,(-(\lambda + \mu)\,t). \tag{7.18}$$

În acest caz disponibilitatea sistemului va fi probabilitatea ca singura lui componentă să fie în funcțiune, adică:

$$D(t) = P_1(t), \tag{7.19}$$

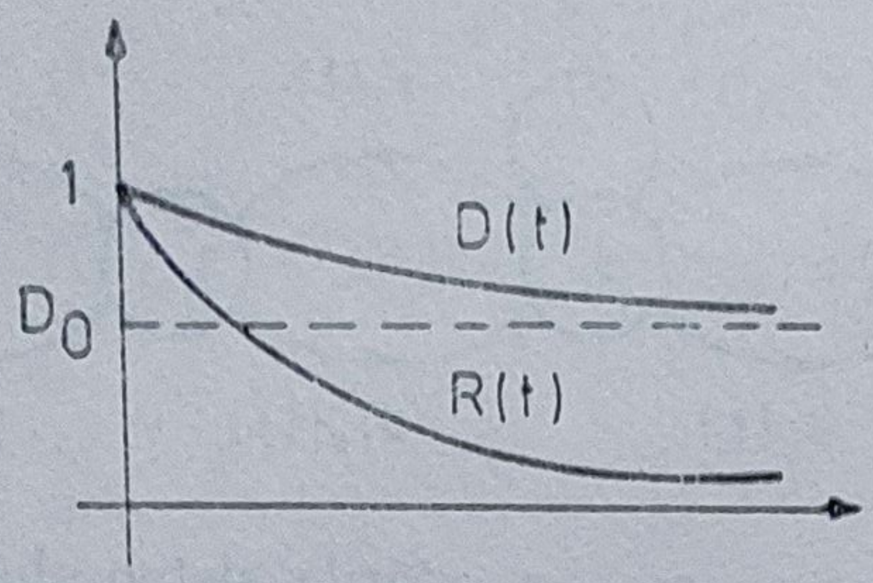


Fig. 7.4. Fiabilitatea și disponibilitatea unui sistem reparabil cu o singură componentă

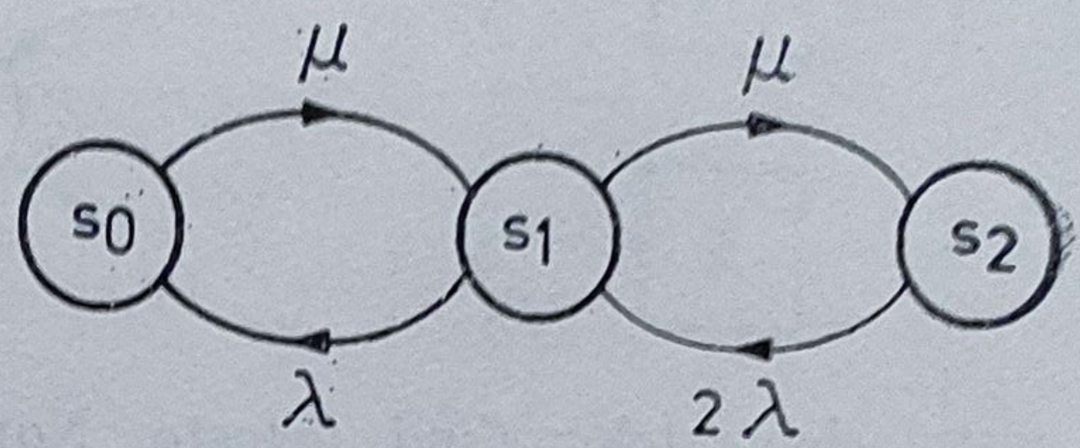


Fig. 7.5. Tranziția stărilor unui sistem redondant cu două componente.

iar coeficienții staționari de disponibilitate și de indisponibilitate sînt:

$$D_0 = \frac{\mu}{\mu + \lambda} = \frac{1/\lambda}{1/\lambda + 1/\mu} = \frac{MTBF}{MTBF + MTTR}; \tag{7.20}$$

$$I_0 = \frac{\lambda}{\mu + \lambda} = \frac{1/\mu}{1/\lambda + 1/\mu} = \frac{MTTR}{MTBF + MTTR}, \tag{7.21}$$

unde $MTTR$ este *media timpului de restabilire*. Evoluția diferită în timp a fiabilității și disponibilității unui asemenea sistem este ilustrată în figura 7.4.

În cazul unui ansamblu de două componente identice, cu redondanță activă totală, presupunînd că ambele componente sînt caracterizate de aceeași intensitate de defectare λ, iar reparațiile sînt executate cu intensitatea μ, graful de tranziție a stărilor sistemului este cel prezentat în figura 7.5.

Rezolvîndu-se sistemul 7.16 pentru acest caz, se determină probabilitățile $P_i(t)$, $i = 0, 1, 2$, iar disponibilitatea staționară va fi:

$$D = \frac{\mu^2 + 2\lambda\mu}{\mu^2 + 2\lambda\mu + 2\lambda^2} = \frac{1 - (1 - D_0)^2}{1 + (1 - D_0)^2} = \frac{1 - I_0^2}{1 + I_0^2}. \tag{7.22}$$

Indisponibilitatea în această situație este:

$$I = \frac{2I_0^2}{1 + I_0^2} < I_0, \quad 0 < I_0 < 1. \tag{7.23}$$

De exemplu, pentru $I_0 = 10\%$ se obține $I \simeq 2\%$.

- **Disponibilitea unui circuit de date**

În scopul analizelor de fiabilitate, sistemele teleinformatice sînt partiționate în subsistemele componente, rezultînd o structură serială de felul celei prezentate în figura 7.1. Notînd cu $D_k(t)$ și $I_k(t)$ disponibilitatea, respectiv indisponibilitatea subsistemului k, atunci pentru întregul sistem vom avea:

$$D(t) = \prod_{k=1}^{n} D_k(t); \tag{7.24}$$

$$I(t) = \sum_{k=1}^{n} I_k(t). \tag{7.25}$$

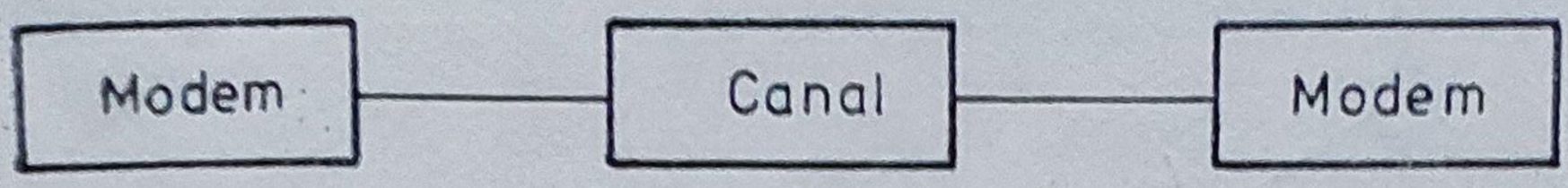


Fig. 7.6. Structura unui circuit de date

Un subsistem pentru rețeaua de transport a datelor poate fi considerat *circuitul de date*, format din trei componente conectate în serie: două modemuri și canalul de comunicație (fig. 7.6).

Notînd cu D_m disponibilitatea staționară a modemurilor, și cu D_c, pe cea a canalului atunci, conform relației 7.24, disponibilitatea circuitului va fi:

$$D = D_m^2 D_c. \tag{7.26}$$

Disponibilitățile staționare D_m și D_c se calculează cu relația 7.20, în care intervin două mărimi: $MTBF$-ul, menționat (uneori) în certificatul produsului și $MTTR$-ul, a cărui valoare depinde de metodologia de semnalizare și înlăturare a deranjamentelor. De exemplu, pentru modem, dacă $MTBF = 10\,000$ ore, iar $MTTR = 10$ ore, atunci obținem $D_m = 0{,}999$ (99,90%).

În cazul liniilor de transmisiuni se ține, de regulă, o statistică a deranjamentelor, precum și a timpilor de remediere, date ce pot servi pentru calculul disponibilității canalului. Valorile acestui parametru diferă mult funcție de tipul liniei; de exemplu, pentru liniile rețelei ARPA coeficientul de indisponibilitate este de ordinul a 10^{-3}. Presupunînd disponibilitatea canalului $D_c = 0{,}99$ (99,00%), atunci conform relației 7.26 se obține pentru circuitul de date $D = 0{,}988$ (98,80%). Vom recurge la tehnica redondării elementelor sistemului pentru a-i îmbunătăți disponibilitatea; inițial vom prevedea componente de rezervă pentru modemuri, apoi pentru canal. Asigurînd pentru fiecare modem cîte o rezervă, conform relației 7.22 se obține o valoare mai bună a disponibilității acestui subansamblu ($D'_m = 99{,}99\%$) și, implicit, o îmbunătățire a disponibilității circuitului de date ($D = 99{,}00\%$). Analog, procedînd la redondarea canalului prin asigurarea unui circuit de rezervă, se obține $D'_c = 99{,}98\%$, iar pentru circuitul de date $D = 99{,}78\%$, fără componente de rezervă pentru modemuri. Din acest exemplu simplu se deduce necesitatea asigurării de rezerve pentru componentele cele mai puțin fiabile, obținîndu-se astfel un cîștig mai mare asupra disponibilității sistemului.

- **Indicatori de fiabilitate specifici STI**

După cum s-a arătat, rețeaua de transport a informației face ca STI să posede unele caracteristici specifice, care trebuie avute în vedere la analiza disponibilității. Se impune deci studiul influenței topologiei rețelei asupra disponibilității STI. În acest scop, literatura de specialitate face apel la modelarea configurației rețelei de transport a datelor cu ajutorul grafurilor liniare, în care nodurile reprezintă punctele de prelucrare a informației, iar arcele — liniile de comunicație dintre procesoare.

În dorința de a asigura lucrării o relativă autonomie, se va face o succintă prezentare a conceptelor fundamentale pentru grafuri (fig. 7.7). Un *graf* G (N, A) este constituit dintr-o mulțime de noduri N și o mulțime de arce A (uneori, în teoria grafurilor aplicată la rețele se folosesc pentru noduri și arce termenii *vîrfuri*, respectiv *muchii*). *Arcul* face legătura dintre două noduri.

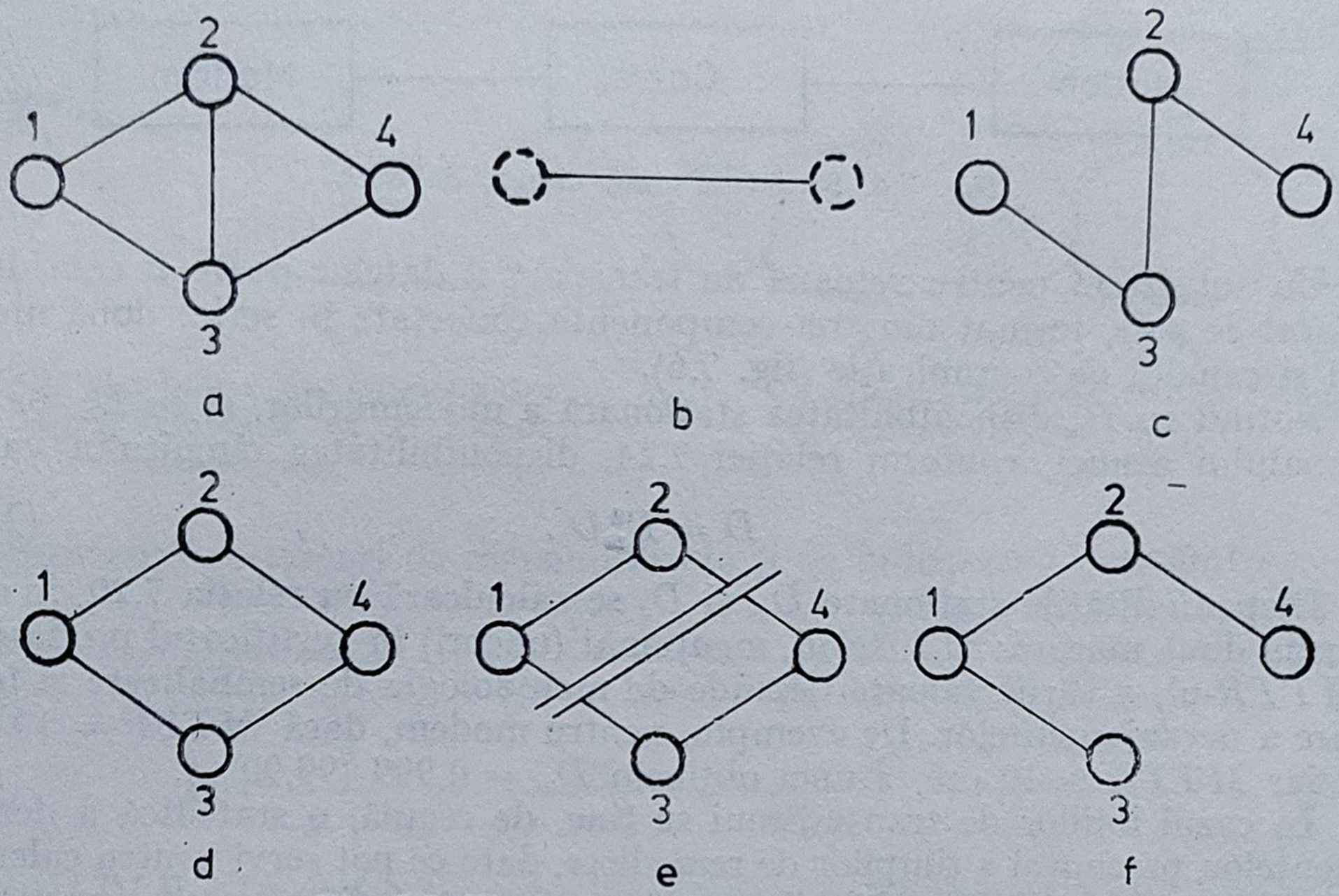


Fig. 7.7. Elemente definitorii pentru grafuri:
a — graf; *b* — arc; *c* — drum; *d* — circuit; *e* — tăietură; *f* — arbore.

Deoarece în cazul STI liniile de legătură sînt bidirecționale, arcele nu vor fi direcționate. Unei linii i se poate asocia o anumită caracteristică, ca de exemplu lungimea, capacitatea maximă de transport etc. Prin definiție, două noduri sau arce sînt *adiacente* dacă au un arc, respectiv un nod, comun. Un *drum (cale)* într-un graf este format dintr-o succesiune de noduri și arce adiacente, iar lungimea drumului este dată de numărul de arce dintre primul și ultimul nod. Între orice pereche de noduri se poate determina, dacă există, un drum de lungime minimă, iar valoarea maximă dintre lungimile tuturor acestor drumuri posibile este *diametrul* grafului. Un drum în care primul nod și ultimul coincid se numește *circuit* care, analog, este caracterizat de o lungime, iar valoarea maximă dintre lungimile tuturor circuitelor minime se numește *circumferința* grafului. Pentru graful din figura 7.7 a, $N = \{1, 2, 3, 4\}$, iar $A = \{(1,2), (1, 3), (2, 1), (2, 3), (2, 4), (3, 1), (3, 2), (3, 4), (4, 2), (4, 3)\}$ și are diametrul 2, iar circumferința 4.

Un graf se numește *conex (conectat)* dacă există cel puțin un drum între oricare pereche de noduri ale sale. Graful conectat fără circuite se numește *arbore.*

Un graf poate fi, de asemenea, descris prin *matricea de adiacență*. Aceasta este o matrice simetrică, avînd numărul de linii egal cu numărul de noduri, ale cărei elemente a_{ij} indică numărul de arce dintre nodul i și nodul j. Pentru graful din figura 7.7a, matricea de adiacență este:

$$A = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix}$$

Cu ajutorul matricei de adiacență se pot analiza proprietățile grafului și determina dacă graful este conectat [D-10].

Deoarece modelul topologic al unui STI este un graf conectat, este important să se analizeze pericolul deconectării grafului în urma căderii unor noduri sau linii, definindu-se indicatorii topologici de fiabilitate conectivitatea și coeziunea. *Conectivitatea* unui graf reprezintă numărul minim de noduri sau linii a căror nefuncționare duce la deconectarea grafului. Prin urmare, acest parametru se poate referi la arce sau la noduri. De exemplu, deconectarea perechii de noduri (1, 4) a grafului din figura 7.7a poate fi provocată de căderea nodurilor 2 și 3, cît și a uneia dintre mulțimile de arce {(1, 2), (1, 3)}, {(2,4), (3, 4)}, {(1, 2), (2, 3), (3, 4)} sau {(1, 3), (2, 3), (2.4)}. O asemenea mulțime de noduri, a căror cădere implică deconectarea grafului, se numește set de noduri de tăiere sau *tăietură de noduri*. Analog se definește și *tăietura de arce*.

Deoarece cazul separării dintr-un graf a unui subgraf cu mai multe noduri este mai nefavorabil decît izolarea unui singur nod, a fost necesar să se introducă un indicator mai general decît conectivitatea. Astfel s-a definit *coeziunea* grafului, reprezentînd numărul minim de arce a căror nefuncționare provoacă izolarea din graf a unui subgraf cu un număr de noduri. Întrucît conectivitatea și coeziunea reprezintă criterii deterministe de fiabilitate pentru rețelele de calculatoare, literatura de specialitate este preocupată atît de evaluarea lor [D-10], cît și de sinteza unor topologii fiabile [P-1]. De asemenea, în [C-4] se definește un indicator topologic general de fiabilitate, care înglobează alături de parametrii menționați și alții, concluzionîndu-se că, pentru obținerea unei structuri cu o fiabilitate ridicată, este necesar ca graful corespunzător rețelei să aibă valori cît mai mari pentru circumferință, conectivitate și coeziune și o valoare cît mai mică pentru diametru. Pe baza indicatorului topologic de fiabilitate general, în lucrare sînt comparate diferite configurații de rețele.

7.2.2. EVALUAREA DISPONIBILITĂȚII STI

Rețeaua de comunicație a unui STI are o structură complexă, reunind echipamente de diferite tipuri și performanțe. O estimare a disponibilității unui asemenea macrosistem numai pe baza topologiei sale este incompletă, fiind necesar să se ia în calcul și fiabilitatea componentelor sale. Se impune, deci, ca modelul topologic al rețelei să fie completat cu indicatorii de fiabilitate probabilistici ai nodurilor și liniilor.

Anterior a fost exemplificată metodologia de calcul a disponibilității pentru un circuit de date, indicator ce va fi asociat în continuare cu probabilitatea de funcționare a liniei. Analog se poate determina această probabilitate pentru nodul de comunicație. În continuare, urmează să se evalueze disponibilitatea întregului sistem, format în exclusivitate din noduri și linii, ținîndu-se seama atît de topologie, cît și de indicatorii probabilistici de fiabilitate determinați anterior.

- **Disponibilitatea rețelelor arborescente**

O arhitectură arborescentă constă din mai multe niveluri ierarhice; pe nivelul superior se situează calculatorul central, iar pe cele inferioare, terminalele utilizator, între acestea existînd mai multe niveluri intermediare ce-

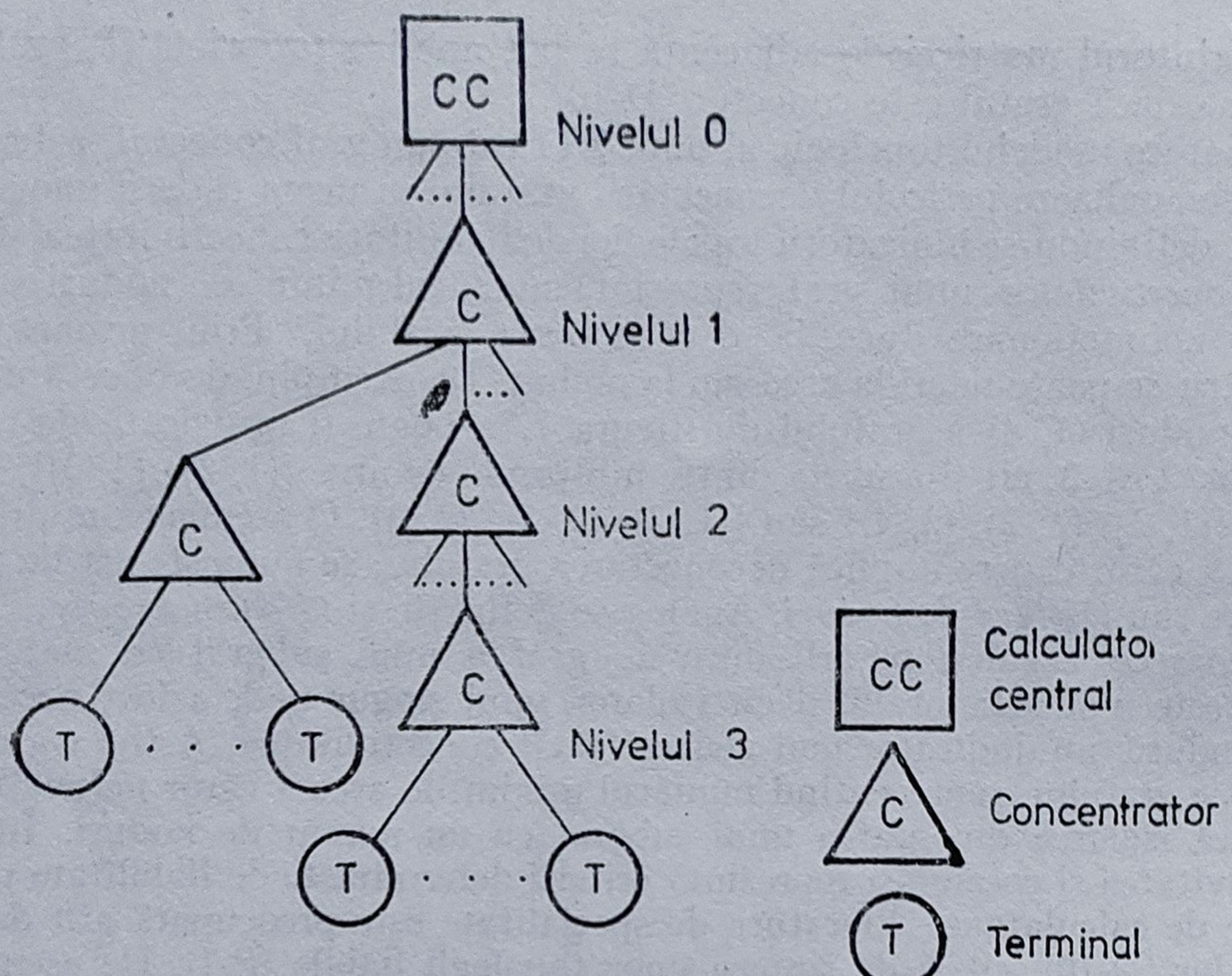


Fig. 7.8. Rețea arborescentă cu patru niveluri

corespund concentratoarelor de date. Asemenea topologii — de altfel puțin fiabile — se adoptă pentru STI mici, specializate. Un exemplu de rețea cu configurație arborescentă avînd patru niveluri este prezentată în figura 7.8. Notînd cu D_k ($k = 0,1, 2, 3$) disponibilitatea nivelului k, atunci — conform relației 7.24 — disponibilitatea corespunzătoare unui terminal de pe nivelul i va fi:

$$D_i = \prod_{k=0}^{i} D_k, \quad i = 1, 2, 3. \tag{7.27}$$

Așadar, cu cît nivelul corespunzător unui utilizator este mai „îndepărtat" de calculatorul central, cu atît legătura sa va fi mai instabilă în timp, iar o întrerupere a legăturii de pe un nivel superior va afecta pe toți utilizatorii pe care-i deservește.

- **Disponibilitatea rețelelor de tip plasă**

Întrucît într-o rețea de calculatoare menținerea legăturii dintre procesoare este de primă importanță, studiile de fiabilitate au fost direcționate în sensul determinării probabilității de comunicare dintre noduri, formulîndu-se următoarele probleme [S-7]:

— problema fiabilității simple — disponibilitatea comunicației dintre o anumită pereche de noduri;

— problema fiabilității parțiale pentru o mulțime de k noduri specificate — disponibilitatea comunicației dintre toate perechile setului respectiv de noduri;

— problema fiabilității generale — disponibilitatea comunicației dintre toate perechile de noduri posibile ale rețelei.

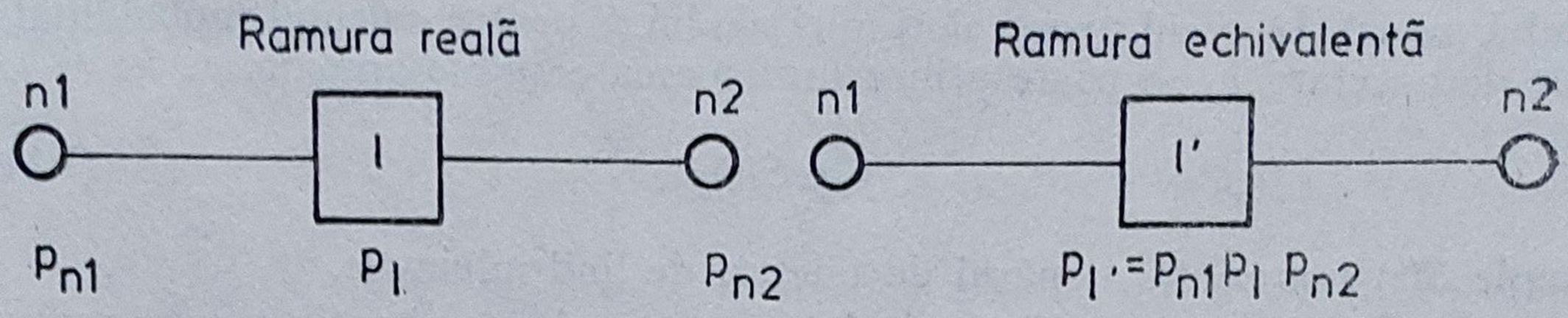


Fig. 7.9. Echivalarea unei ramuri reale

Pentru simplificare calculele se fac, de regulă, în ipoteza că atît liniile, cît și nodurile se defectează și sînt reparate independent. Cu toate că, în multe cazuri, această ipoteză este falsă, totuși ea nu cauzează erori intolerabile în evaluarea indicatorilor de fiabilitate [S-8]. De asemenea, tot pentru simplificare, de multe ori se are în vedere numai căderea liniilor, nodurile fiind considerate perfect fiabile sau invers. În realitate însă atît nodurile, cît și liniile unui sistem de comunicație au anumite probabilități de defectare, constatîndu-se preocupări pentru îmbunătățirea unor algoritmi bazați pe asemenea ipoteze. Astfel, în [A-2] se propune o metodă de extindere a algoritmilor elaborați în ipoteza unor noduri nedefectabile, care se bazează pe observația: defectarea unui nod implică și căderea liniilor incidente lui. Prin urmare, orice *ramură* (linie, împreună cu nodurile adiacente) poate fi echivalată cu o combinație serie de nod perfect și linie cu o probabilitate modificată de cădere. Deoarece o linie presupune o legătură bidirecțională, ambele noduri de la extremitățile sale vor trebui luate în considerare în relația de echivalare. În figura 7.9 se descrie această operație; de asemenea, relația de calcul a probabilității liniei echivalente, unde cu p_{n1}, p_{n2} și p_l s-au notat probabilitățile de defectare a nodurilor, respectiv a liniei.

În cazul rețelelor de tip plasă majoritatea modelelor elaborate vizează rezolvarea fie a problemei fiabilității simple — determinarea probabilității deconectării unei anumite perechi de noduri, fie a celei generale — calculul probabilității deconectării întregii rețele.

Pentru o rețea cu n linii care au aceeași probabilitate de defectare p, în ipoteza că nodurile sînt perfect fiabile, probabilitatea ca sistemul să se afle în starea cu k linii defecte este:

$$P(k) = C_n^k p^k (1-p)^{n-k}, \quad k = 0,1, \ldots, n. \tag{7.28}$$

În acest caz probabilitatea deconectării rețelei p_d va fi:

$$p_d = \sum_{k=c}^{n} S(k)\, p^k (1-p)^{n-k}, \tag{7.29}$$

unde cu $S(k)$ s-a notat numărul de stări cu k linii defecte, stări în care rețeaua este deconectată, valoarea ce satisface condițiile:

$$1 \leqslant S(k) \leqslant C_n^k, \quad (\forall)\ k = c,\ c+1, \ldots, n. \tag{7.30}$$

Cu alte cuvinte, $S(k)$ reprezintă numărul tăieturilor de k linii care deconectează rețeaua, iar c este chiar conectivitatea corespunzătoare liniilor gra-

fului rețelei. În cazul unor valori mici ale lui p, pentru calculul probabilității de deconectare p_d se poate utiliza următoarea expresie aproximativă:

$$p_d \simeq S(c)p^c, \tag{7.31}$$

unde $S(c)$ reprezintă numărul de tăieturi de linii minimale.

În [D-10] se exemplifică calculul exact și cel aproximativ pentru probabilitatea p_d, iar în [P-1] sînt date relațiile echivalente pentru calculul probabilității de deconectare în cazul cînd liniile sînt perfect fiabile și toate nodurile au aceeași probabilitate de cădere.

În general, modelele de evaluare a disponibilității bazate pe examinarea tuturor tăieturilor sau a căilor dintre perechile de noduri sînt prohibite, chiar și pentru calculatoarele rapide (s-a demonstrat că pentru o rețea cu n noduri și l linii, numărul tăieturilor este de ordinul 2^{n-2}, iar a căilor 2^{1-n+2}).

În scopul reducerii numărului de căi și a tăieturilor care urmează să fie examinate s-au încercat diferite metode aproximative. De exemplu, în [J-1] se arată că pentru analizele de fiabilitate nu toate căile unui graf trebuie analizate, ci numai *căile minimale de bază*, unice pentru un anumit graf, iar în [A-3] se propune o metodă de minimizare a numărului de căi analizate bazată pe *arborii de traversare* (spanning trees) ai grafului.

Pentru determinarea probabilității de deconectare dintre o pereche de noduri mai eficiente par a fi metodele aproximative bazate pe descompunerea sistematică a grafului probabilistic al rețelei în două părți printr-o tăietură adecvată. O asemenea tehnică, prezentată în [A-4], s-a dovedit a fi economicoasă, în ceea ce privește timpul de calcul, chiar și pentru rețelele mari. Metoda este generală și furnizează, pe lîngă valoarea numerică a disponibilității, și expresia ei analitică, ca funcție de probabilitățile de defectare a liniilor, nodurile fiind considerate perfecte. Vom exemplifica aplicarea acestei metode pentru rețeaua al cărei graf fiabilistic este prezentat în figura 7.10a, în scopul determinării probabilității de deconectare dintre nodul 1 (sursa) și nodul 5 (terminal) conform *algoritmului ABC* (denumire datorată autorilor).

Pasul 1: Se alege tăietura de arce $\{b, c, d\}$ care descompune graful în subgrafurile I și II. Subgraful I va conține ramurile acestei tăieturi și cele care sînt de aceeași parte cu nodul sursă, iar subgraful II va conține restul de ramuri (fig. 7.10b).

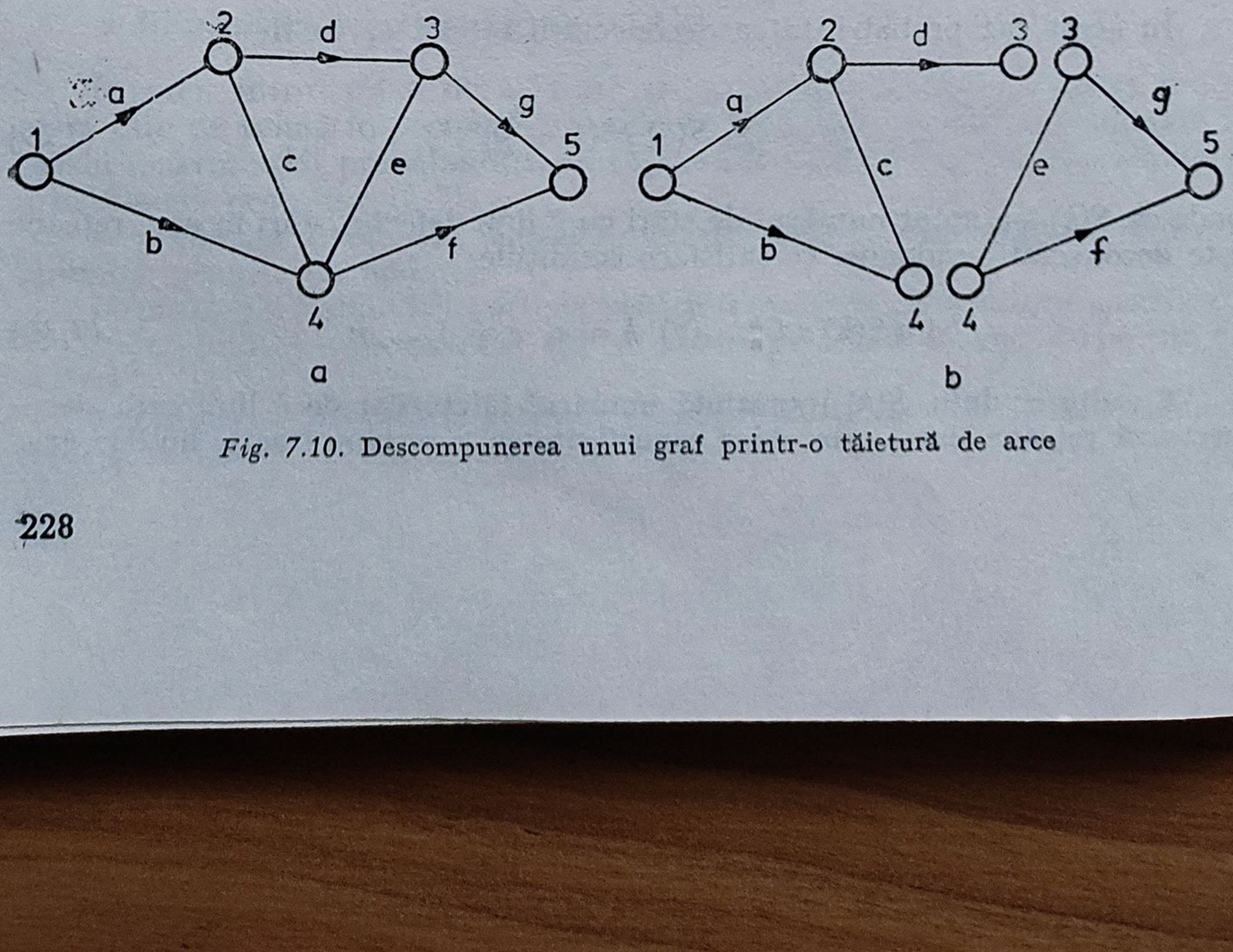


Fig. 7.10. Descompunerea unui graf printr-o tăietură de arce

Pasul 2: Se scrie mulțimea de noduri N, ordonată astfel:

$$N = \{n_1,\ n_2, \ldots, n_k,\ n_{k+1}, \ldots, n_{k+m}, \ldots, n_t\},$$

unde n_1 și n_t reprezintă nodurile sursă, respectiv terminal, iar $N_c = \{n_k, n_{k+1}, \ldots, n_{k+m}\}$ — este mulțimea de noduri comune celor două subgrafuri. În cazul nostru subgraful I are $k = 2$ noduri, iar $N_c = \{3,\ 4\}$.

Pasul 3: Se scrie matricea de adiacență C pentru graful unitar și pentru cele două subgrafuri (D, respectiv E):

$$C = \begin{pmatrix} 0 & a & 0 & b & 0 \\ 0 & 0 & d & c & 0 \\ 0 & 0 & 0 & e & g \\ 0 & c & e & 0 & f \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \qquad D = \begin{pmatrix} 0 & a & 0 & b & 0 \\ 0 & 0 & d & c & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & c & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \qquad E = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & e & g \\ 0 & 0 & e & 0 & f \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Pasul 4: Se determină toate drumurile dintre nodul sursă n_1 și nodurile n_j, $j = k,\ k+1, \ldots, k+m$, asociindu-le evenimentele de succes N_j [A-3]. Pentru subgraful I aceste căi sînt următoarele: de la n_1 la n_3 — ad și bcd, $N_3 = \mathrm{AD} + \mathrm{BCD}$; de la n_1 la n_4 — b și ac, $N_4 = B + \mathrm{AC}$.

Pasul 5: Analog, se determină toate drumurile de la nodurile n_j la n_t, pentru $j = k,\ k+1, \ldots, k+m$, cărora li se asociază evenimentele de succes M_j. Pentru subgraful II avem următoarele căi: de la n_3 la n_5 — ef și g, $M_3 = \mathrm{EF} + \mathrm{G}$; de la n_4 la n_5 — eg și f, $M_4 = \mathrm{EG} + \mathrm{F}$.

Pasul 6: Se partiționează mulțimea de noduri comune N_c în două submulțimi disjuncte în toate modurile posibile, obținîndu-se submulțimile $N_{c_{1,i}}$ și $N_{c_{2,i}}$ pentru $i = 1,\ 2, \ldots, 2^m$. În cazul nostru $m = 2$ și:

$N_{c_{1,1}} = \{3,\ 4\}$, $\qquad N_{c_{2,1}} = \{\varnothing\}$;

$N_{c_{1,2}} = \{4\}$, $\qquad N_{c_{2,2}} = \{3\}$;

$N_{c_{1,3}} = \{3\}$, $\qquad N_{c_{2,3}} = \{4\}$;

$N_{c_{1,4}} = \{\varnothing\}$, $\qquad N_{c_{2,4}} = \{3,\ 4\}$.

Pasul 7: Se calculează probabilitățile parțiale de deconectare a subgrafurilor I și II, $p_{\mathrm{I},i}$ respectiv $p_{\mathrm{II},i}$ cu relațiile:

$$p_{\mathrm{I},i} = P\{X_i\},\quad X_i = \bigcap_{j \in N_{c_{1,i}}} \mathrm{N_j} \bigcap_{j \in N_{c_{2,i}}} \overline{\mathrm{N}}_\mathrm{j}; \tag{7.32}$$

$$p_{\mathrm{II},i} = 1 - \mathrm{P}\{\mathrm{Y_i}\},\quad \mathrm{Y_i} = \bigcup_{j \in N_{c_{1,j}}} \mathrm{M_j}, \tag{7.33}$$

unde $P\{X_i\}$ reprezintă probabilitatea evenimentului X_i.

Probabilitatea deconectării perechii de noduri $(\mathrm{n_s},\ \mathrm{n_t})$ va fi:

$$\mathrm{I_{s-t}} = \sum_{i=1}^{2^m} p_{\mathrm{I},i} p_{\mathrm{II},i} \tag{7.34}$$

Considerînd pentru liniile rețelei din figura 7.10a următoarele probabilități de nefuncționare: $\mathrm{p_a} = 0{,}1$; $\mathrm{p_b} = 0{,}4$; $\mathrm{p_c} = 0{,}3$; $\mathrm{p_d} = 0{,}1$; $\mathrm{p_e} = 0{,}1$; $\mathrm{p_f} = 0{,}4$ și $\mathrm{p_g} = 0{,}3$, rezultatele obținute pentru acest caz sînt date în tabelele 7.1 și 7.2.

TABELUL 7.1 **Rezultatele parțiale pentru subgraful I**

i	X_i	X_i disjuncte	$P\{X_i\}$	$P\{X_i\}$ numerice
1	$(AD + BCD) \times (B + AC)$	$ABD + \bar{A}BCD + A\bar{B}CD$	$q_aq_bq_d + p_aq_bq_cq_d + q_ap_bq_cq_d$	0,7506
2	$(B + AC) \times \overline{(AD + BCD)}$	$B\bar{D} + A\bar{B}C\bar{D} + \bar{A}B\bar{C}D$	$q_bp_d + q_ap_bq_cp_d + p_aq_bp_cq_d$	0,1004
3	$(AD + BCD) \times \overline{(B + AC)}$	$A\bar{B}\bar{C}D$	$q_ap_bp_cq_d$	0,0972
4	$\overline{(AD + BCD)} \times \overline{(B + AC)}$	$\bar{A}\bar{B} + \bar{A}B\bar{C}\bar{D}$	$p_ap_b + q_ap_bp_cp_d$	0,0508

TABELUL 7.2 **Rezultatele parțiale pentru subgraful II**

i	Y_i	Y_i disjuncte	$P\{Y_i\}$	$1-P\{Y_i\}$ numerice
1	$(EF + G) + (EG + F)$	$G + F\bar{G}$	$q_g + q_fp_g$	0,120
2	$F + EG$	$F + E\bar{F}G$	$q_f + q_ep_fq_g$	0,148
3	$EF + G$	$G + EF\bar{G}$	$q_g + q_eq_fp_g$	0,138
4	$\varnothing$	$\varnothing$	0	1,000

Conform relației 7.34, expresia analitică a indisponibilității corespunzătoare perechii de noduri (1,5) va fi:

$$I_{1-5} = (q_aq_bq_d + p_aq_bq_cq_d + q_ap_bq_cq_d)(1 - q_g - q_fp_g) +$$
$$+(q_bp_d + q_ap_bq_cp_d + p_aq_bp_cq_d)(1-q_f-q_ep_fq_g)+q_ap_bp_cq_d(1 - q_g - q_eq_fp_g)+$$
$$+ (p_ap_b + q_ap_bp_cp_d),$$

iar pentru disponibilitate se va obține următoarea valoare numerică:

$$D_{1-5} = 1 - I_{1-5} = 0{,}8307072.$$

Exemplul a fost rezolvat și pentru alte tăieturi de descompunere a grafului, obținîndu-se același rezultat, care a fost verificat și prin alte metode.

O îmbunătățire a acestei metode este propusă în [R-4]. Noutatea constă în simplificarea grafului prin reduceri serie și paralel, înainte de a fi descompus. Această tehnică s-a dovedit eficientă chiar și pentru rețele foarte mari. Algoritmul modificat, denumit *algoritmul R-ABC*, este de asemenea ușor de implementat pe calculator.

7.3. EVALUAREA TIMPULUI DE RĂSPUNS ȘI A ÎNCERCĂRII RESURSELOR

Timpul de răspuns și încercarea sistemului sînt performanțe de bază pentru STI; prima prezintă importanța majoră pentru utilizator, iar ce-a de-a doua pentru gestionarul sistemului. În cazul aplicațiilor teleinformatice există

o adevărată „competiție" între timpul de răspuns, care se urmărește să fie cît mai mic, și încărcarea sistemului, dorită cît mai eficientă. În cadrul analizei timpului de răspuns se va urmări evaluarea componentelor acestui parametru: întîrzierea datorată transportului datelor, adică timpul de tranzit și timpul necesar prelucrării informației.

Timpul de tranzit se datorează în principal așteptărilor la care este supus mesajul în bufferele nodurilor de joncțiune. Dacă se dorește luarea în calcul și a timpilor de propagare pe linii, atunci aceștia vor fi cumulați la întîrzierile din nodurile de incidență.

Coeficientul de încărcare se va estima pentru următoarele resurse: canalele de legătură, procesoarele pentru prelucrarea datelor și bufferele din noduri.

7.3.1. METODOLOGIA ESTIMĂRII PERFORMANȚELOR

- **Evaluarea performanțelor unei legături de date**

Considerăm două noduri adiacente A și B, între care are loc un transfer de informație, transmiterea realizîndu-se de la A către B. Sistemul format din cele două noduri și canalul de legătură poate fi reprezentat sub forma unui *model cu cozi de așteptare*, ca în figura 7.11. Capacitatea canalului se presupune de C biți/s și se fac următoarele ipoteze privind funcționarea stației de servire: (I1) — capacitatea bufferului din nodul B este nelimitată, astfel că toate mesajele sosite sînt acceptate; (I2) — sistemul este perfect fiabil; (I3) — nodul sursă are capacitatea infinită, iar mesajele sosesc aleator, sub forma unui proces poissonian de parametru λ (în medie λ mesaje/s). Notînd cu L lungimea mesajelor, variabilă pe care o presupunem de asemenea distribuită după o lege exponențială, atunci și timpul de servire a unui mesaj va fi repartizat tot după o lege exponențială de parametru C/L. În aceste ipoteze probabilitatea staționară ca în sistem să existe n mesaje, notată $P(n)$, verifică relația 7.35, care rezultă din graful de tranziție a stărilor sistemului prezentat în figura 7.12.:

$$(\lambda + C/L)P(n) = C/LP(n+1) + \lambda P(n-1), \quad \text{pentru} \quad n \geqslant 1. \qquad (7.35)$$

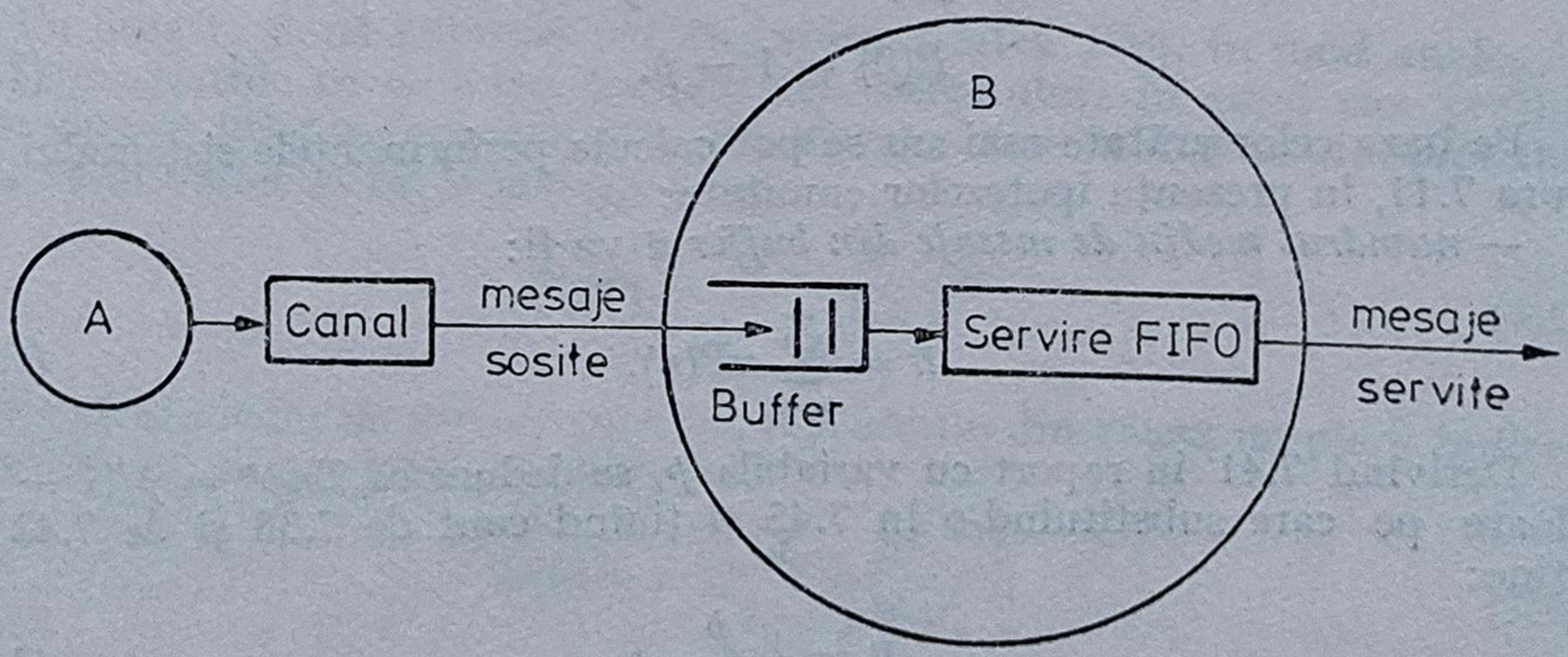


Fig. 7.11. Modelul unui sistem de transmisie unidirecțională dintre două noduri

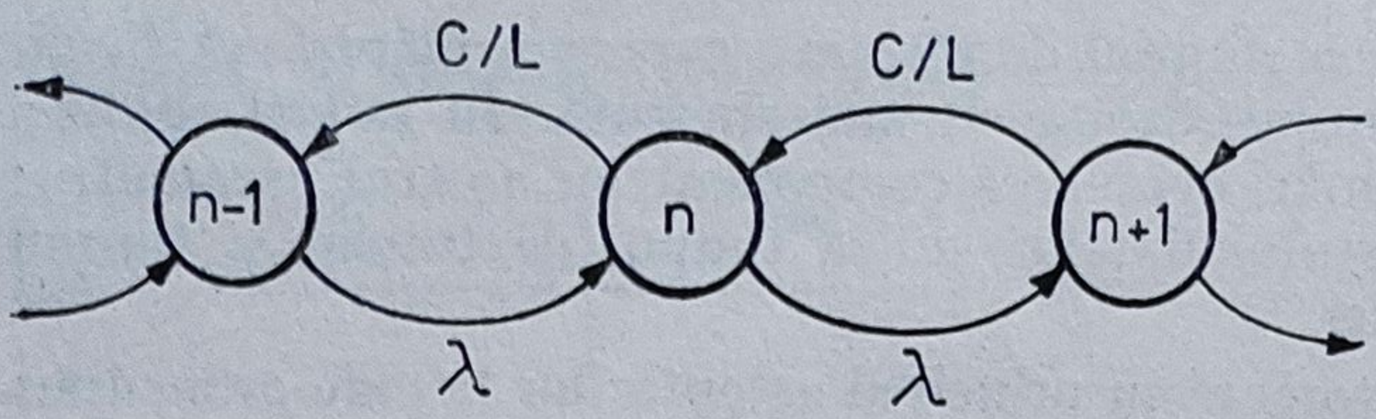


Fig. 7.12. Graful de tranziție a stărilor stației de servire

Scriind relația 7.35 pentru începutul procesului, rezultă:

$$\lambda P(0) = C/LP(1) \quad \text{sau:} \tag{7.36}$$

$$\mathrm{P}(1) = \frac{\lambda}{C/L} P(0). \tag{7.37}$$

În ultima relație, cîtul $\frac{\lambda}{C/L}$ redă raportul dintre numărul de mesaje sosite și capacitatea de servire a stației, care se numește *intensitate de trafic*, și se notează cu p. Prin recurență, din 7.35 și 7.36 rezultă:

$$P(n) = p^n P(0) \tag{7.38}$$

Probabilitățile $P(n)$ trebuie să satisfacă relațiile:

$$\sum_{n=0}^{\infty} P(n) = 1, \quad \text{adică:} \tag{7.39}$$

$$P(0) \sum_{n=0}^{\infty} p^n = 1. \tag{7.40}$$

Seria 7.40 este convergentă dacă $\mathrm{p} < 1$, adică $\frac{\lambda}{\mathrm{C/L}} < 1$. Prin urmare, pentru ca sistemul să fie stabil este necesară o capacitate de servire superioară frecvenței medii a sosirilor. În aceste condiții rezultă:

$$\sum_{n=0}^{\infty} p^n = \frac{1}{1-p}; \tag{7.41}$$

$$P(0) = 1 - p. \tag{7.42}$$

Pe baza celor arătate mai sus se pot calcula performanțele sistemului din figura 7.11, în prezența ipotezelor enunțate:

— *numărul mediu de mesaje din buffer* $\bar{n}$ va fi:

$$\bar{n} = \sum_{n=0}^{\infty} n\mathrm{P}(n). \tag{7.43}$$

Derivînd 7.41 în raport cu variabila p, se deduce că $\Sigma np^n = p/(1-p)^2$, valoare pe care substituind-o în 7.43 și ținînd cont de 7.38 și de 7.42 se obține:

$$\bar{n} = \frac{p}{1-p}; \tag{7.44}$$

— *timpul mediu de traversare a nodului* $\overline{T}$:

Variabila T are două componente — timpul de așteptare în coadă și timpul de servire, astfel că:

$$\overline{T} = \bar{n}\frac{1}{C/L} + \frac{1}{C/L}. \tag{7.45}$$

Ținînd cont de 7.44 și de expresia variabilei ρ, se obține:

$$\overline{T} = \frac{1}{C/L - \lambda}. \tag{7.46}$$

De asemenea, dacă în 7.45 înlocuim pe $\bar{n}$ cu valoarea din 7.44 și ținînd seama că $L/C = \rho/\lambda$, obținem binecunoscuta relație a lui Little:

$$\overline{T} = \frac{\bar{n}}{\lambda}; \tag{7.47}$$

— *coeficientul de utilizare a canalului* va fi dat de probabilitatea ca el să fie activ, adică:

$$P(n > 0) = 1 - P(0) = \rho. \tag{7.48}$$

Modelul prezent este idealist, întrucît în practică capacitatea bufferelor este limitată, așa că vom proceda la estimarea acestor performanțe într-o situație mai realistă, modificînd prima ipoteză : (I'1) capacitatea bufferului din nodul B este M, iar toate mesajele ce sosesc după umplerea lui sînt pierdute (fenomenul de *blocare*).

În această situație, din condiția

$$\sum_{n=0}^{M} P(n) = 1 \tag{7.49}$$

obținem:

$$P(0) = \frac{1 - \rho}{1 - \rho^{M+1}}. \tag{7.50}$$

Cu această nouă expresie pentru probabilitatea $P(0)$, în mod analog, se pot scrie relațiile ce definesc comportarea sistemului. În acest caz prezintă de asemenea interes probabilitatea de blocare, care este dată de probabilitatea ca în buffer să fie M mesaje, adică:

$$P(M) = \frac{1 - \rho}{1 - \rho^{M+1}}\rho^{M}, \tag{7.51}$$

iar numărul mediu de mesaje care vor fi pierdute din cauza umplerii bufferului va fi dat de produsul $\lambda P(M)$:

$$\lambda P(M) = \lambda\frac{1 - \rho}{1 - \rho^{M+1}}\rho^{M}. \tag{7.52}$$

Să luăm acum în calcul și defectările canalului, reformulînd a doua ipoteză: (I'2) canalul este supus la defecte, a căror repartiție în timp este descrisă

de un proces poissonian de parametru α, iar timpii necesari pentru restabilire sînt distribuiţi tot după o lege exponenţială negativă de medie β^{-1}. Evident, un coeficient de indisponibilitate al canalului nenul implică degradarea performanţelor. Considerînd capacitatea bufferului $M = \infty$, atunci timpul mediu de traversare a unui nod va fi:

$$\bar{T} = \frac{L/C + \delta^2\alpha/\beta}{\delta - \lambda L/C}, \tag{7.53}$$

unde $\delta = \beta/(\beta + \alpha)$. Valoarea $\bar{T}$ este mai mare decît în cazul unui canal perfect fiabil (δ este indisponibilitatea canalului). Coeficientul de utilizare a canalului în acest caz va fi $\delta\rho$, parametru mai mic decît în ipoteza (I2).

- **Evaluarea timpului de răspuns şi a încercării unui procesor de prelucrare**

Ne propunem să evaluăm performanţele de utilizare pentru o altă resursă a STI — procesorul, destinat prelucrării datelor. Cum într-un STI se vehiculează de regulă mai multe tipuri de lucrări, este necesar să se adopte o anumită disciplină în prelucrarea datelor, astfel ca în lipsa unor priorităţi să nu se favorizeze lucrările de o anumită categorie. O asemenea disciplină, frecvent utilizată, este prelucrarea în cuante de timp. Cuanta este timpul de servire alocat unei lucrări într-o secvenţă, iar dacă la sfîrşitul secvenţei lucrarea este terminată, ea va părăsi sistemul, altfel va fi repusă în vîrful cozii de aşteptare. Să considerăm un procesor dintr-un STI care prelucrează într-o asemenea manieră lucrări de două categorii, *C1* şi *C2*, de caracteristici esenţial diferite (de exemplu, clasa *C1* este formată din loturi de lucrări pentru terminalele grele, iar *C2* sînt lucrările pentru terminalele conversaţionale). Sosirile în sistem ale lucrărilor sînt considerate de tip Poisson de parametri λ_1 şi λ_2, iar timpii medii de prelucrare sînt $1/\mu_1$ pentru lucrările *C1* şi $1/\mu_2$ pentru lucrările *C2*, valori ce diferă substanţial. Figura 7.13 prezintă modelul utilizat pentru evaluarea performanţelor unui asemenea sistem.

Probabilitatea ca la un moment dat în coada de aşteptare a sistemului să existe n_1 lucrări din clasa *C1* şi n_2 din clasa C_2 va fi [M-1]:

$$P(n_1, n_2) = \left[1 - \left(\frac{\lambda_1}{\mu_1} + \frac{\lambda_2}{\mu_2}\right)\right]\left(\frac{\lambda_1}{\mu_1}\right)^{n_1}\left(\frac{\lambda_2}{\mu_2}\right)^{n_2}\frac{(n_1 + n_2)!}{n_1! n_2!} \tag{7.54}$$

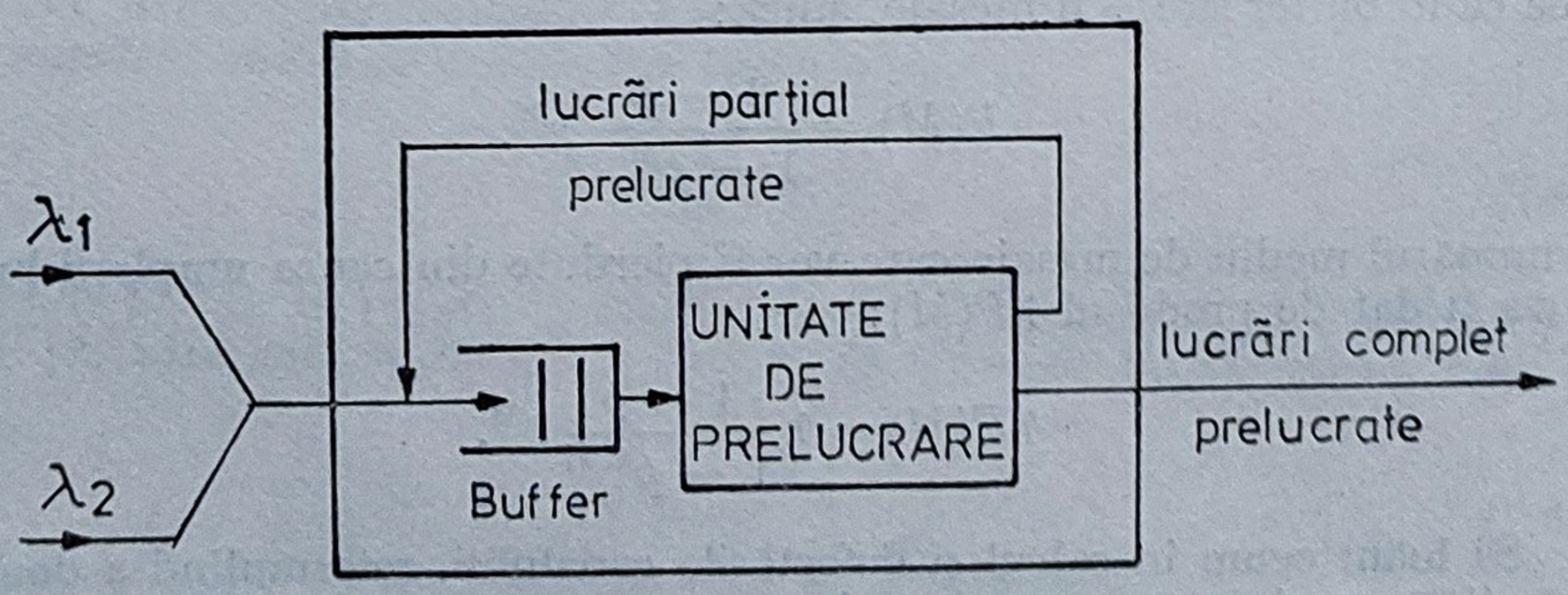


Fig. 7.13. Modelul matematic al unui sistem de prelucrare

Notînd cu $\bar{n}_1$ și $\bar{n}_2$ numărul mediu de lucrări din buffer din clasa *C1*, respectiv *C2*, se obține:

$$\bar{n}_1 = \frac{\lambda_1/\mu_1}{1-(\lambda_1/\mu_1+\lambda_2/\mu_2)}; \quad \bar{n}_2 = \frac{\lambda_2/\mu_2}{1-(\lambda_1/\mu_1+\lambda_2/\mu_2)}. \tag{7.55}$$

Cu ajutorul formulei lui Little 7.47 se pot calcula timpii medii de răspuns ai sistemului $\bar{T}_1$, respectiv $\bar{T}_2$:

$$\bar{T}_1 = \frac{1/\mu_1}{1-(\lambda_1/\mu_1+\lambda_2/\mu_2)}; \quad \bar{T}_2 = \frac{1/\mu_2}{1-(\lambda_1/\mu_1+\lambda_2/\mu_2)}. \tag{7.56}$$

Timpul mediu de răspuns pentru întregul sistem va fi:

$$\bar{T} = \frac{\bar{n}_1+\bar{n}_2}{\lambda_1+\lambda_2} = \frac{\lambda_1/\mu_2+\lambda_2/\mu_2}{(\lambda_1+\lambda_2)[1-(\lambda_1/\mu_1+\lambda_2/\mu_2)]}. \tag{7.57}$$

Coeficientul de utilizare al procesorului este dat de probabilitatea ca el să fie în lucru, adică:

$$P(n_1>0 \text{ sau } n_2>0) = 1-P(0,0) = \lambda_1/\mu_1+\lambda_2/\mu_2. \tag{7.58}$$

7.3.2. INFLUENȚA PROTOCOALELOR DE COMUNICAȚIE ASUPRA PERFORMANȚELOR STI

Într-un STI se vehiculează pe lîngă blocuri de date și informație de serviciu, specifică protocoalelor de legătură de pe diferite niveluri, ceea ce va duce la diminuarea performanțelor sistemului în ansamblu. În scopul evidențierii comportării unei legături de date se va examina transportul bidirecțional pe ambele canaluri, luînd în considerare atît mesajele informaționale cît și confirmările, pentru două protocoale de comunicație punct-la-punct TMM-UC și HDLC. Cu ajutorul modelului cu cozi de așteptare va fi urmărită comportarea sistemului din figura 7.14a, format din două noduri adiacente legate prin două canale de transmisie, în prezența ipotezelor (I1)—(I3) formulate anterior.

- **Protocol TMM-UC**

Transmisia datelor în cazul protocolului TMM-UC se face prin confirmarea individuală a fiecărui mesaj informațional primit de către receptor, astfel că după ce un mesaj a fost emis o nouă emisie va fi posibilă numai după primirea confirmării destinatarului. La un moment dat un canal al sistemului din figura 7.14a se va putea afla într-una din stările: D — transmisia unui mesaj de date; I — inactiv; C_0 — transmisia unui mesaj de confirmare. Sistemul, analizat pe baza stărilor celor două canale, va avea o comportare descrisă de graful din figura 7.14b. Se observă că stările (D, C_0) și (C_0, D) sînt excluse, întrucît dacă pe un canal este în curs o transmisie de date, confirmarea recepției se va putea face numai după recepționarea completă și verificarea corectitudinii mesajului respectiv.

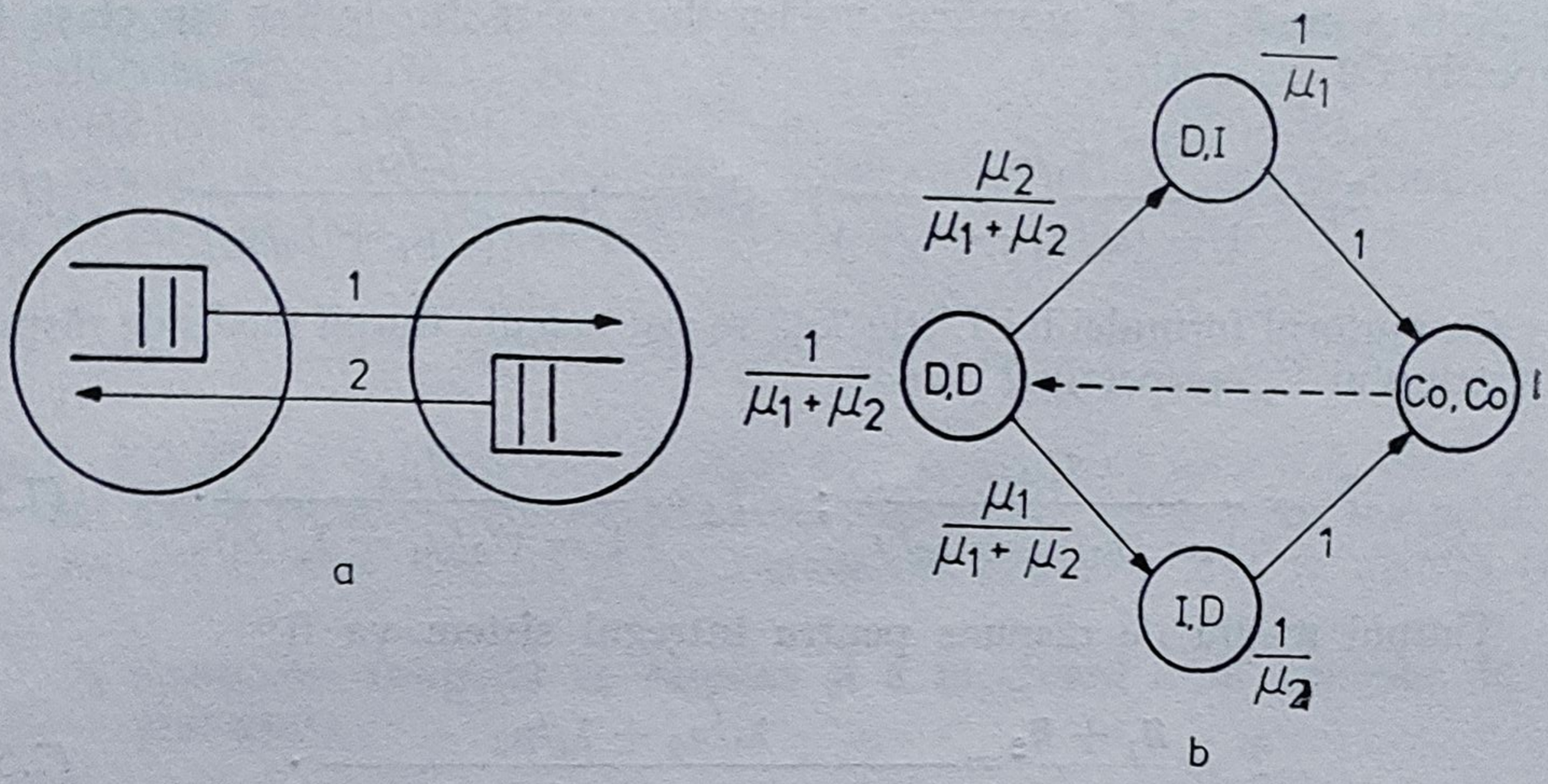


Fig. 7.14. Comportarea unei legături de date sub protocol TMM-UC

Presupunem că lungimile pachetelor de date transmise pe cele două canale sînt repartizate după legi exponențiale de valori medii L_1, respectiv L_2, iar mesajele de confirmare le vom considera de aceeași lungime, l. Dacă capacitatea canalelor este aceeași, C, atunci și timpii de transmisie a datelor vor fi repartizați tot după legi exponențiale, ale căror valori medii sînt $\mu_1^{-1} = L_1/C$, respectiv $\mu_2^{-1} = L_2/C$. Pentru estimarea performanțelor legăturii se vor asocia celor patru stări ale sistemului probabilitățile staționare $P\{(D, D)\}$, $P\{(D, I)\}$, $P\{(I, D)\}$ și $P\{(C_0, C_0)\}$. În scopul simplificării calculelor se va normaliza viteza de transmisie, presupunînd-o egală cu unitatea în ambele sensuri. În această ipoteză, notînd cu k_1 și k_2 încărcarea canalelor, obținem:

$$k_1 = P\{(D, D)\} + P\{(D, I)\}; \tag{7.59}$$
$$k_2 = P\{(D, D)\} + P\{(I, D)\}.$$

iar încărcarea legăturii de date, notată k, va fi:

$$k = k_1 + k_2. \tag{7.60}$$

În ipotezele enunțate timpul de coincidență a transmisiilor de date pe cele două canale va fi tot o variabilă aleatoare de medie $(\mu_1 + \mu_2)^{-1}$, iar probabilitatea stării (D, D), proporțională cu această valoare. Din graful probabilistic prezentat în figura 7.14b se obțin următoarele probabilități pentru stările sistemului:

$$P\{(D, D)\} = \alpha \frac{1}{\mu_1 + \mu_2};$$

$$P\{(D, I)\} = \alpha \frac{\mu_2}{\mu_1(\mu_1 + \mu_2)}; \tag{7.61}$$

$$P\{(I, D)\} = \alpha \frac{\mu_1}{\mu_2(\mu_1 + \mu_2)};$$

$$P\{(C_0, C_0)\} = \alpha l,$$

unde α este constanta de normalizare, a cărei valoare rezultă din condiția:

$$P\{(D, D)\} + P\{(D, I)\} + P\{(I, D)\} + P\{(C_0, C_0)\} = 1, \qquad (7.62)$$

adică:

$$\alpha = \frac{\mu_1\mu_2(\mu_1 + \mu_2)}{\mu_1\mu_2(\mu_1 + \mu_2)\,l + \mu_1\mu_2 + \mu_1^2 + \mu_2^2}. \qquad (7.63)$$

Cu ajutorul relației 7.60 se obține coeficientul pentru încărcarea circuitului:

$$k = \frac{\mu_1^{-1} + \mu_2^{-1}}{l + \mu_1^{-1} + \mu_2^{-1} - (\mu_1 + \mu_2)^{-1}}. \qquad (7.64)$$

Dacă se neglijează timpul de transmisie a confirmărilor ($l = 0$) și se presupune că mesajele de date sînt de lungimi egale în cele două sensuri ($L_1 = L_2$), se obține $k = 4/3$. Dar cum acest coeficient redă încărcarea maximă a întregului circuit, format din două canale, pentru un singur canal rezultă coeficientul $k = 2/3$, adică 66%. Prin urmare, pentru o legătură de date gestionată de protocolul TMM-UC coeficientul de încărcare maxim nu poate depăși 66% din capacitatea canalelor.

Testarea modelului s-a făcut prin măsurători asupra unor linii de capacitate $C = 12\,200$ biți/s, modificîndu-se lungimile blocurilor de date L_1 și L_2 [G-2]. O parte din rezultatele experimentale obținute pentru o lungime a mesajelor pe canalul 1 constantă sînt prezentate în tabelul 7.3.

TABELUL 7.3 **Încărcările maxime teoretice (k_1, k_2, k) și măsurate (k_1', k_2', k') pentru o legătură gestionată de protocolul TMM-UC. $L_1 = 37$ octeți, $l = 7$ octeți**

L_2 [octeți]	36	67	117	177	227
k_1 [biți/s]	11 366	7 400	5 345	3 733	2 968
k_1' [biți/s]	12 000	8 000	5 100	3 600	2 800
k_2 [biți/s]	11 366	15 401	16 906	17 849	18 225
k_2' [biți/s]	12 000	14 700	16 250	16 800	17 500
k [biți/s]	22 732	22 801	22 251	21 582	21 193
k' [biți/s]	24 000	22 700	21 350	20 400	20 300

Măsurătorile au permis validarea modelului teoretic, care descrie bine comportarea sistemului real. Abaterile se datorează ipotezelor simplificatoare în prezența cărora a fost elaborat modelul și faptului că, în realitate, lungimea blocurilor de confirmare a fost nenulă. Se poate concluziona că protocolul TMM-UC nu permite o bună exploatare a capacității liniilor, fapt constatat de altfel și pentru alte protocoale de legătură.

- **Protocolul HDLC**

Pe o legătură de date, gestionată de protocolul HDLC, se transmit mesaje pe tipurile *I*, *S* și *U*; mesajele de tip *I* sînt informaționale, cele de tip *S* de supervizare, iar mesajele de tip *U* realizează controlul și modificarea parametrilor legăturii. Confirmările se fac cu mesajele de tipul *I* sau *S* și nu cu

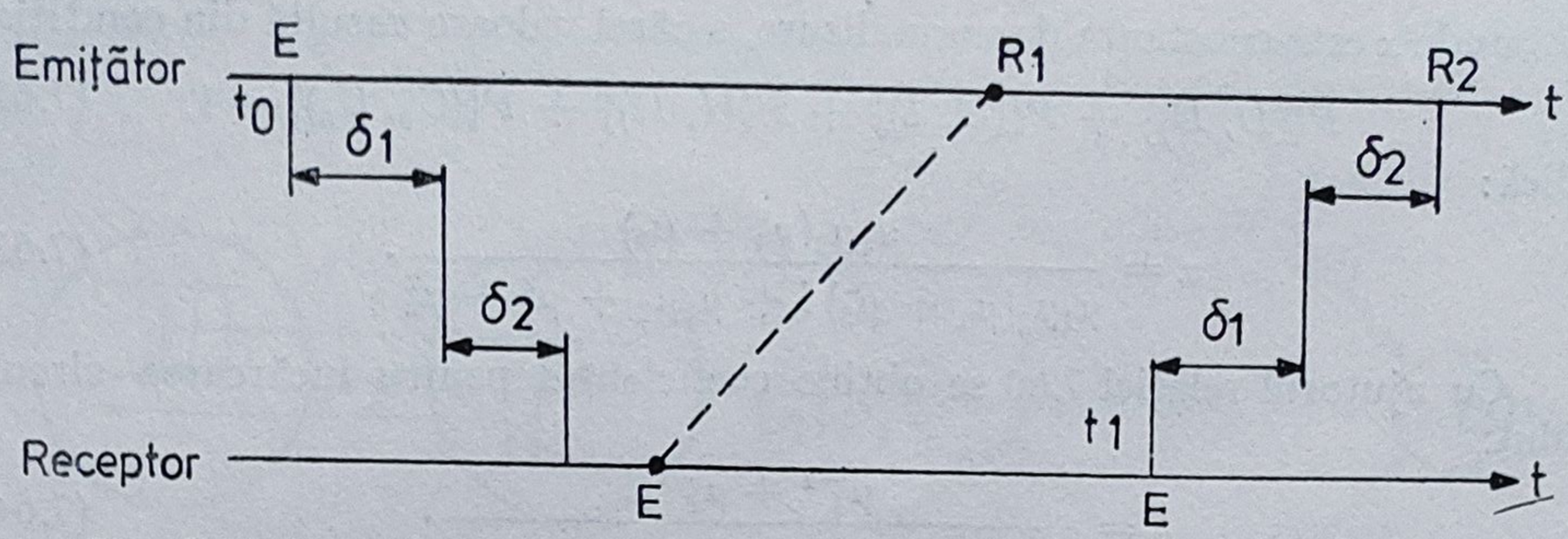


Fig. 7.15. Momentele transmisiei unui mesaj în cazul protocolului HDLC

mesaje speciale care să realizeze numai această funcție ca în cazul procedurii TMM-UC. De asemenea, nu este necesar să se mențină un sincronism între emițător și receptor în dialogul emisie-confirmare, confirmările — pozitive sau negative — putînd fi individuale sau colective. Totuși, numărul maxim de mesaje pe care un nod le poate avea recepționate și neconfirmate este limitat de capacitatea bufferului său. Această tehnică de dialog este menită să aducă o îmbunătățire a utilizării canalelor de transmisie.

Analiza se va face în lipsa mesajelor de tip U, situîndu-ne într-un regim de lucru staționar, în prezența ipotezelor enunțate la începutul paragrafului. Să urmărim cronograma din fig. 7.15 care redă momentele semnificative ale emisiei, recepției și confirmării unui mesaj. Fie t_0 momentul emisiei mesajului. La receptor datele vor fi disponibile după întîrzierea δ_1, datorată recepției și prelucrărilor. În continuare, în timpul δ_2, se verifică corectitudinea mesajului recepționat. De menționat că întîrzierile δ_1 și δ_2 sînt constante. Totuși, confirmarea recepției nu este posibil să fie făcută cu mesajul proxim emis de către receptor după momentul $t_0 + (\delta_1 + \delta_2)$, întrucît acesta a fost deja pregătit și nu mai poate fi modificat. Ea este posibilă abia cu următorul mesaj, emis la momentul t_1, și va fi disponibilă la emițător nu mai devreme de cea de-a doua recepție (R_2) de la momentul t_0, după o întîrziere mai mare decît constanta 2 $(\delta_1 + \delta_2)$. În [L-5] este prezentat modelul matematic cu cozi de așteptare ce descrie comportarea legăturii sub protocolul HDLC, în care se utilizează următorii parametri: L_1, L_2 — lungimile medii ale mesajelor vehiculate de la emițător către receptor, respectiv în sens invers; C — capacitatea canalului de transmisie; N — mărimea bufferului unui nod; μ_1^{-1}, μ_2^{-1} — timpii medii de transmisie a unui mesaj pe canalul 1, respectiv 2($\mu_1^{-1} = L_1/C$ și $\mu_2^{-1} = L_2/C$). Întrucît modelul, în forma lui inițială, nu poate fi rezolvat, pentru evaluarea performanțelor legăturii s-a formulat un model simplificat (fig. 7.16), în care mai multe stații au fost înlocuite cu una singură de parametru $S = 2(\delta_1 + \delta_2) + 2/\mu_2$. Media timpului de servire, S, are următoarele componente: $2(\delta_1 + \delta_2)$ pentru prelucrarea mesajelor emise la momentele t_0 și t_1; $1/\mu_2$ pentru transmiterea mesajului emis imediat după $t_0 + \delta_1 + \delta_2$ și $1/\mu_2$ pentru transmisia mesajului emis la momentul t_1.

În cazul sistemului cu saturație astfel definit, încărcarea maximă k este dată de relația:

$$k = \mu_1 \left(1 - \frac{\rho^N/N!}{1 + \rho + \rho^2/2! + \ldots + \rho^N/N!}\right), \tag{7.65}$$

unde $\rho = \mu_1 S$.

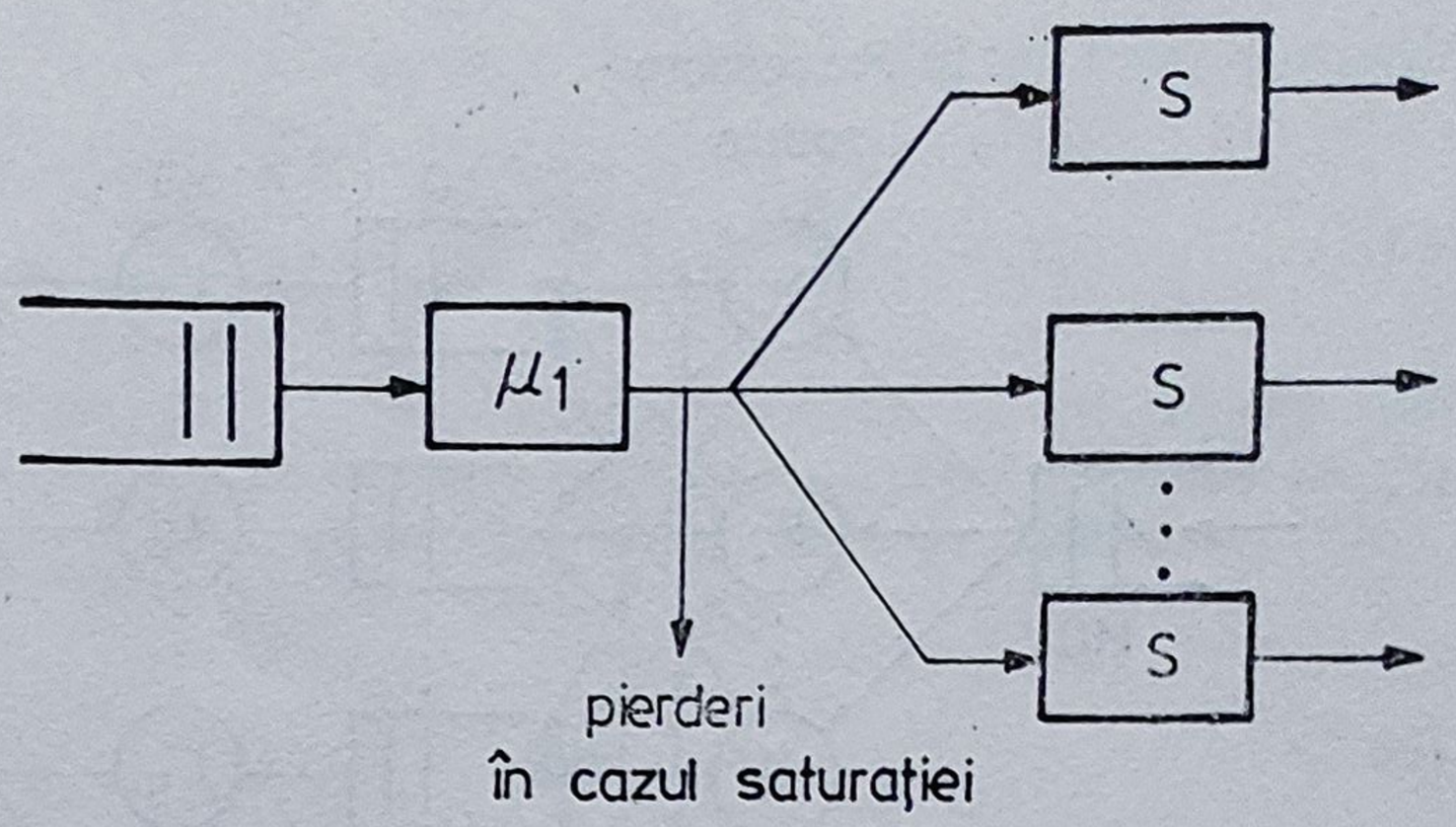


Fig. 7.16. Modelul matematic simplificat pentru o legătură de date sub protocolul HDLC

Media timpului de servire, $\overline{T}$, este dată de raportul:

$$\overline{T} = \frac{k}{\mu_1}. \tag{7.66}$$

Rezultatele obținute pe modelul simplificat au fost comparate cu măsurătorile efectuate pe rețeaua CYCLADES, pe linii cu capacitatea de 4 800, 19 200 și 48 000 biți/s pentru mesaje cu lungimile de 300 și 1 000 biți. Datele experimentale au permis validarea modelului teoretic, evidențiind totodată o exploatare mai eficientă a capacității canalelor decît în cazul protocolului TMM-UC.

În scopul îmbunătățirii performanțelor în cazul protocolului HDLC au fost propuse și unele soluții. Una dintre acestea vizează reducerea întîrzierii confirmărilor prin modificarea structurii mesajului, în sensul plasării octetului de comandă — care conține și codul confirmării — înaintea informației de control. Prin plasarea confirmării cît mai spre sfîrșitul mesajului, acesta ar putea fi actualizat chiar pe durata emisiei, putîndu-se confirma în acest fel chiar și ultima recepție.

7.3.3. ANALIZA PERFORMANȚELOR REȚELELOR DISTRIBUITE

O rețea distribuită cuprinde calculatoare de prelucrare a datelor (denumite și procesoare *host*) și rețeaua de comunicație. Comunicarea în rețea este asigurată de către procesoare *front-end*, denumite interfețe de comunicație (communication interfaces, prescurtat *CI*). Un asemenea *CI* este de regulă un minicalculator mono sau multiprocesor, destinat să rezolve toate problemele de comunicație: controlul fluxului, gestiunea bufferelor liniilor, detectarea și corectarea erorilor, etc.

Pentru analiza performanțelor rețelelor distribuite s-au făcut diverse studii bazate pe teoria cozilor de așteptare, dar — în general — modelele respective au fost elaborate în prezența ipotezelor simplificatoare (I1)—(I3) enunțate la începutul paragrafului. De asemenea, întîrzierile din rețea au fost atribuite în exclusivitate comportării canalelor de comunicație, nu și a procesoarelor nodale. În realitate, întîrzierea unui mesaj într-un *CI* se

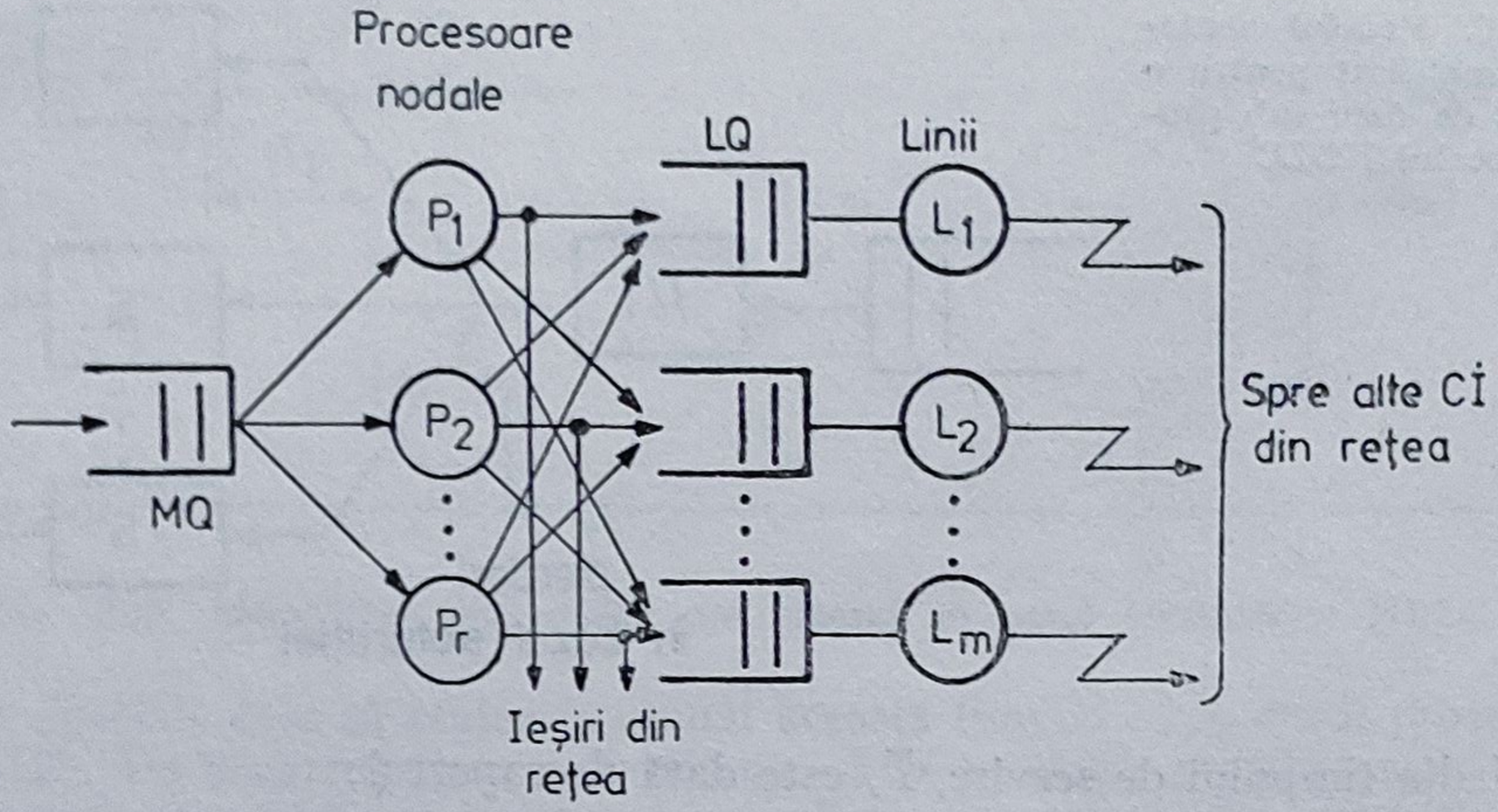


Fig. 7.17. Modelul analitic al unui CI

datorează atît așteptării pentru un canal liber, cît și timpului de așteptare în coada procesorului.

În continuare se va indica metodologia de evaluare a timpului de tranzit într-o rețea distribuită, luîndu-se în considerare și întîrzierile datorate prelucrărilor din noduri. Pentru fiecare *CI* se construiesc un sistem compus dintr-o stație de tip $M/D/r$ și mai multe stații $M/M/1$ (fig. 7.17). Intrările în sistem pot proveni de la procesoarele host conectate la *CI*-ul respectiv (intrări externe) sau de alte *CI*-uri din rețea (intrări interne). La sosirea unui mesaj în sistem, acesta va fi introdus într-un buffer de intrare (message queue, prescurtat *MQ*). Toate bufferele sînt considerate de capacitate nelimitată, iar sosirea mesajelor în sistem se presupune repartizată după o lege exponențială. Mesajele din *MQ* sînt preluate după disciplina first-come—first-serve (primul sosit, primul servit), de către unul dintre procesoare care realizează toate prelucrările necesare. Întrucît toate mesajele, indiferent de mărime, necesită aceleași prelucrări, timpul de servire va fi o constantă pentru un anumit *CI*. După prelucrare, mesajul va fi plasat în bufferul unei linii (line queue, prescurtat *LQ*), pentru a fi transmis unui alt *CI*, sau va părăsi rețeaua cînd destinația mesajului este unul dintre host-urile deservite de către CI-ul respectiv. Dacă, în mod analog, pentru fiecare nod al rețelei se construiește modelul matematic corespunzător, rețeaua de comunicație devine o rețea deschisă de cozi de așteptare, iar timpul de tranzit al unui mesaj va fi dat de suma întîrzierilor din nodurile străbătute. Presupunem rețeaua formată din n noduri, iar pentru fiecare nod i definim următoarele mărimi medii: λ_i — frecvența sosirii mesajelor în nodul i, considerînd atît intrările interne cît și cele externe; γ_i — frecvența intrărilor externe în nodul i; λ_{ij} — frecvența intrărilor în bufferul LQ_j al nodului i; T_i — timpul total de așteptare al mesajului în bufferul MQ al nodului i, incluzîndu-se și timpul de prelucrare; T_{ij} — timpul total de așteptare în bufferul LQ_j al nodului i, incluzîndu-se și timpul de transmisie a mesajului.

Traficul extern total din rețea va fi dat de suma:

$$\gamma = \sum_{i=1}^{n} \gamma_i. \tag{7.67}$$

Notînd cu m_i numărul canalelor incidente în nodul i, timpul mediu de traversare a rețelei, T, va fi:

$$T = \frac{1}{\gamma}\left[\sum_{i=1}^{n}\left(\lambda_i T_i + \sum_{j=1}^{m_i}\lambda_{ij}T_{ij}\right)\right]. \tag{7.68}$$

În scopul determinării mărimii λ_i este necesar ca pentru fiecare nod să se estimeze și intensitatea traficului ce provine din rețea. Notînd cu p_{jk} probabilitatea ca un mesaj generat la nodul j să fie destinat nodului k, intensitatea traficului intern pentru nodul i este:

$$\sum_{\substack{j=1\\ j\neq i}}^{n}\ \sum_{k\in căii\ (j\to k,\,i)}^{n} p_{jk}\gamma_j, \tag{7.69}$$

unde produsul $p_{jk}\gamma_j$ reprezintă ponderea fluxului unei căi dintre nodurile (j, k), care trece prin CI_i, la intrarea internă a nodului i. În aceste condiții, fluxul total incident în nodul i este:

$$\lambda_i = \gamma_i + \sum_{\substack{j=1\\ j\neq i}}^{n}\ \sum_{k\in căii\ (j\to k,\,i)}^{n} p_{jk}\gamma_j. \tag{7.70}$$

Notînd cu δ_i timpul mediu de prelucrare a unui mesaj în nodul i și cu r_i numărul de procesoare al nodului, atunci timpul mediu total de întîrziere datorat stației $M/D/r$ este dat de relația:

$$T_i = \overline{T}_{MQ_i} + \delta_i, \tag{7.71}$$

unde $\overline{T}_{MQ_i}$ reprezintă timpul mediu de așteptare în bufferul MQ_i. Analog, timpul mediu de întîrziere într-o stație $M/M/1$ va fi:

$$T_{ij} = \overline{T}_{LQ_{ij}} + \delta_{ij}, \tag{7.72}$$

în care factorul δ_{ij} este timpul de transmisie a mesajului pe linia j, incidentă nodului i. De asemenea, în literatura de specialitate se dau și relațiile de calcul pentru $\overline{T}_{MQ}$ și $\overline{T}_{LQ}$ — parametri caracteristici stațiilor de tip $M/D/r$, respectiv $M/M/1$. Valoarea medie $\overline{T}_{MQ}$ depinde funcțional de numărul procesoarelor r și de coeficientul de utilizare a acestora, p. Pentru un nod i, probabilitatea p_i este:

$$p_i = \frac{\lambda_i \delta_i}{r_i}. \tag{7.73}$$

În calculul valorii $\overline{T}_{LQ}$ intervine coeficientul de încărcare a canalului respectiv. Pentru linia j a nodului i, acest coeficient este:

$$p_{ij} = \lambda_{ij}\delta_{ij}. \tag{7.74}$$

Substituind 7.71 și 7.72 în 7.68 se obține:

$$T = \frac{1}{\gamma}\sum_{i=1}^{n}\left[\lambda_i(\overline{T}_{MQ_i} + \delta_i) + \sum_{j=1}^{m_i}\lambda_{ij}(\overline{T}_{LQ_{ij}} + \delta_{ij})\right]. \tag{7.75}$$

Această relație de calcul al timpului mediu de traversare a rețelei referă atît caracteristicile canalelor de comunicație, cît și cele ale procesoarelor no-

dale. Totuși, formula 7.75 se bazează pe ipoteza unui trafic poissonian în rețea, prezumție care nu este valabilă îndeosebi în cazul unei încercări ridicate a procesoarelor nodale, astfel că modelul prezentat va furniza date mai puțin exacte chiar pentru situațiile cînd rețeaua va fi supraîncărcată — momente critice ce trebuie controlate cu precădere.

În [S-6] se propune o îmbunătățire a modelului prezentat, introducîndu-se unele corecții care au în vedere îndeosebi aceste situații.

Cînd coeficientul de utilizare a procesoarelor tinde către valoarea maximă, frecvența mesajelor poate fi considerată cunoscută, nemaifiind aleatoare. În primul rînd, această „regularizare" a traficului se produce la nivelul intrărilor în stațiile M/M/1, impunîndu-se o corecție a timpului mediu de așteptare în bufferele LQ. Coeficientul de corecție propus în cazul bufferului j din nodul i este:

$$\alpha_{ij} = 1 - \frac{p_i(1 - p_{ij})\,\lambda_{ij}}{\lambda_i}, \tag{7.76}$$

iar timpul mediu corectat de așteptare în bufferul LQ_{ij} va fi:

$$\overline{T}'_{LQ_{ij}} = \alpha_{ij}\overline{T}_{LQ_{ij}}. \tag{7.77}$$

Comportarea comparativă a celor două modele — corectat și necorectat — a fost evidențiată prin simulare pe calculator. Testarea s-a făcut prin modificarea parametrilor rețelei: numărul de noduri, topologia, numărul procesoarelor din noduri, încărcarea canalelor și a rețelei în general. În tabelul 7.4. sînt prezentate datele comparative pentru o rețea cu trei noduri, complet conectată, obținute prin modificarea numărului de procesoare, încărcările fiind considerate constante.

TABELUL 7.4

Nr. procesoare, r	Coeficientul de utilizare a procesoarelor, p	Timpul mediu de traversare		Eroare relativă [%]	$\overline{T}_{MQ}$ [ms]
		corectat $\overline{T}'$[ms]	necorectat $\overline{T}$[ms]		
1	0,940	54,79	51,91	5,54	15,6
2	0,470	24,83	25,94	4,47	0,3
3	0,313	24,39	25,45	4,34	0,046
4	0,235	24,34	25,44	4,51	0,0078

Se observă că diferența procentuală dintre rezultatele furnizate de cele două metode crește pe măsură ce încărcarea procesoarelor se mărește. Modelul prezentat împreună cu simulatorul său constituie unelte eficiente pentru proiectarea STI mari.

7.4. OPTIMIZAREA STI

În fiecare etapă de elaborare a STI proiectantul trebuie să-și propună implementarea soluției celei mai bune. În cazul proiectării, a optimiza înseamnă, în esență, utilizarea cu eficiență maximă a disponibilităților în vederea reunirii lor într-un sistem. Deși resursele disponibile sînt caracterizate

de anumiți parametri, ele vor trebui combinate în așa fel, încît să se obțină de la sistem anumite funcțiuni și o funcționalitate la un cost cît mai mic. În cazul nostru, componentele care urmează să fie reunite pentru constituirea STI sînt: liniile de teletransmisie; modemurile; concentratoarele și multiplexoarele de date; procesoarele de prelucrare a datelor; software-ul de utilizare (baze de date, programe) ș.a., fiecare dintre acestea fiind caracterizată de anumite performanțe și un anumit cost. Construirea sistemului se face în prezența unor constrîngeri de ordin economico-geografic, topologic sau de altă natură, urmărindu-se respectarea anumitor cerințe de proiectare, cum ar fi:

— asigurarea unei anumite capacități de transport și prelucrare;
— timpul mediu de răspuns să nu depășească o anumită limită;
— să se asigure o anumită funcționalitate pentru sistem etc.

De regulă, performanța care se optimizează — funcția obiectiv a problemei — este costul sau timpul de răspuns, celelalte cerințe intervenind ca restricții. Problema generală de optimizare, astfel formulată, este complexă și dificil de rezolvat, dar cum în practică pot interveni și alte restricții simplificatoare, optimizarea trebuie să aibă în vedere în primul rînd aceste cerințe specifice. Optimizarea poate fi rezolvată prin diferite metode, sintetizate în [D-10], dintre care cele euristice — deși nu conduc la cea mai bună soluție — sînt preferabile datorită timpului de calcul rezonabil.

În activitatea de proiectare trebuie rezolvate deopotrivă probleme ce se referă la întregul sistem cît și „de amănunt"; cele globale fiind abordate, de regulă, în următoarea ordine:

— alegerea topologiei rețelei de transport al informației;
— alegerea capacității liniilor și alocarea fluxului pe fiecare linie;
— stabilirea tehnicii de vehiculare a informației în sistem, precum și a protocoalelor de teletransmisie;
— distribuirea software-ului utilizator în sistem și repartiția sarcinilor de prelucrare pe procesoare.

Deciziile locale se referă la mărimea bufferelor, tipurile de concentratoare și procesoare, lungimea și structura mesajului informațional etc. Pentru abordarea unora dintre problemele menționate vom considera pentru rețeaua de transport următoarele mărimi caracteristice:

— topologia rețelei, descrisă de graful $G(N, A)$;
— fiecare arc a_{ij}, caracterizat de: o capacitate, C_{ij}; o lungime fizică, l_{ij}; un cost q_{ij}, ca funcție de mai multe variabile $q_{ij}=f(l_{ij}, C_{ij},$ calitatea liniei ș.a.). Costul liniei poate îngloba și costul modemurilor care o deservesc, precum și al altor componente;
— timpul mediu necesar pentru a transporta o unitate de informație de-a lungul liniei (i, j), notat T_{ij}, ca o funcție $T_{ij}=f(C_{ij}$ ș.a. parametri);
— matricea de trafic $R=(r_{ij})$, unde r_{ij} reprezintă numărul de unități de informație ce urmează să fie transmise de la nodul i la nodul j în unitatea de timp.

7.4.1. OPTIMIZAREA TOPOLOGIEI

În cazul rețelelor cu o configurație simplă ori în prezența unor restricții impuse de către utilizator sau de către rețeaua de teletransmisie pe care urmează să se grefeze sistemul nu se pune problema unei optimizări topologice.

Dimpotrivă, pentru rețelele mari, cu o largă întindere geografică, unde există multiple posibilități de interconectare a resurselor și pentru care costul transmisiei informației poate avea o pondere considerabilă în costul total al sistemului, alegerea unei topologii care să satisfacă în condiții cît mai avantajoase cerințele de transport devine o necesitate.

Problema topologiei rețelei se pune în primul rînd atunci cînd se optează pentru un anumit tip de arhitectură (stea, arbore, plasă etc.), apoi — odată tipul ales — urmează să se realizeze optimizarea propriu-zisă.

Algoritmii utilizați la construirea rețelei de cost minim pot fi optimali sau euristici. Metodele optimale sînt costisitoare din punctul de vedere al timpului de calcul, motiv pentru care sînt preferate cele euristice. Acestea dau soluții apropiate de cea optimă într-un timp de calcul relativ scurt.

• Optimizarea topologiei rețelelor arborescente

Problemele esențiale ce trebuie rezolvate în cazul proiectării unei rețele arborescente sînt următoarele: cîte concentratoare sînt necesare și unde urmează să fie plasate; care terminale urmează să fie conectate la un anumit concentrator. Aceste sarcini — evident și altele — urmează să fie soluționate prin construirea unui arbore cu un set cît mai mic de legături, în prezența unor restricții legate de capacitatea canalelor, de numărul de terminale suportate de către un anumit concentrator, precum și de caracteristicile terminalelor, respectîndu-se cerințele de trafic, timp de răspuns și siguranță în funcționare.

Restricția pentru fiabilitatea rețelei poate fi formulată prin impunerea unui graf de conectivitate doi (existența a cel puțin două căi între terminale și calculatorul central) sau prin limitarea numărului de terminale (concentratoare) deconectate în urma căderii oricărei linii.

Proiectarea topologiei unei rețele arborescente presupune, în esență, rezolvarea a două probleme (fig. 7.18): conectarea terminalelor la concentratoare și a concentratoarelor la procesorul de prelucrare, ceea ce implică soluționarea următoarelor sarcini [M-1]:

— determinarea numărului de concentratoare;
— localizarea concentratoarelor și a calculatorului central;
— conectarea concentratoarelor la procesor prin linii de viteză mare;

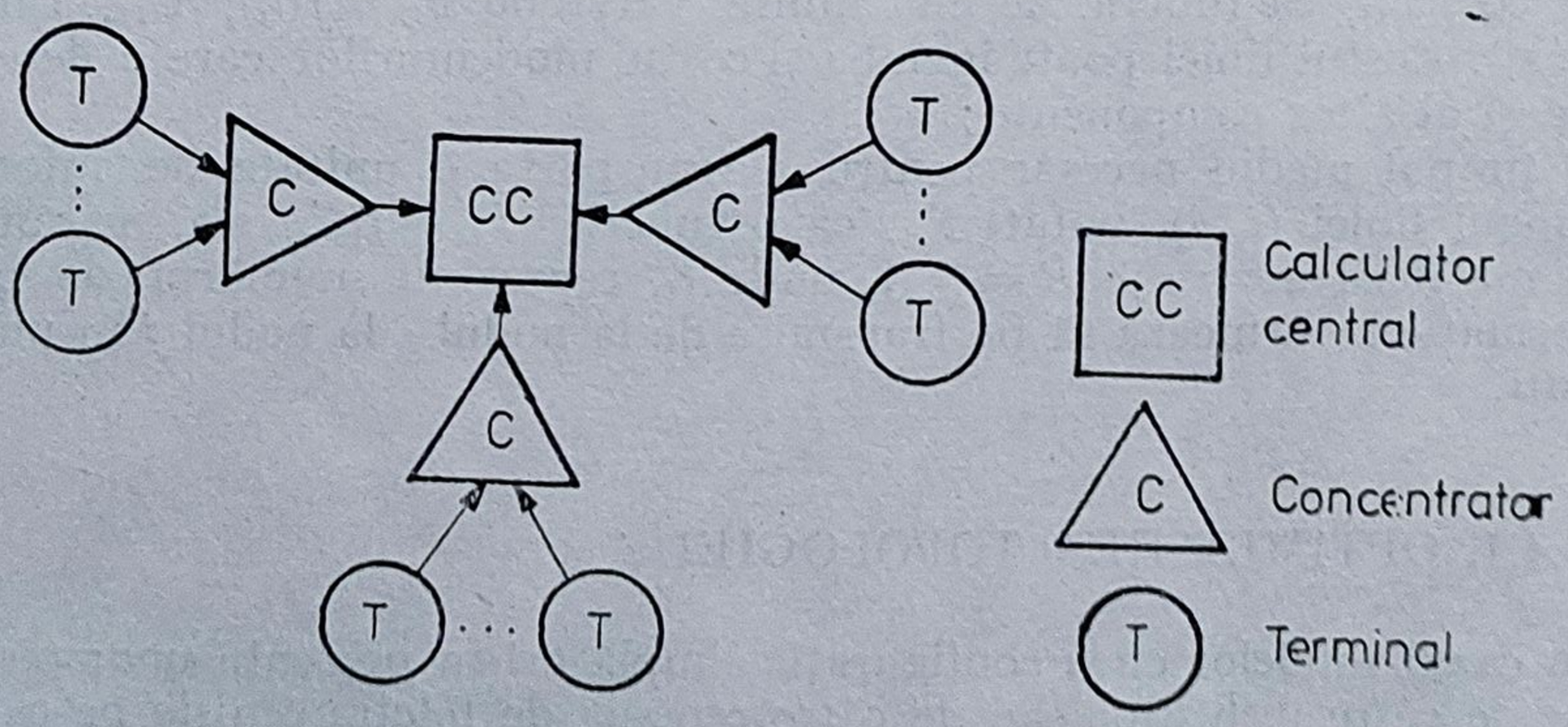


Fig. 7.18. Proiectarea topologiei unei rețele arborescente

— gruparea terminalelor pe concentratoare;
— conectarea terminalelor la concentratoare prin linii de viteză joasă.

Pentru optimizarea topologiilor arborescente au fost elaborați diverși algoritmi — majoritatea euristici —, care rezolvă una sau mai multe dintre problemele enunțate. Întrucît atît conectarea concentratoarelor, cît și a terminalelor necesită realizarea de conexiuni multipunct, sînt utilizați algoritmii pentru generarea arhitecturilor arbore de cost minim în prezența unor restricții, dintre care cei mai cunoscuți sînt algoritmii Prim [P-4], Kruskal [K-5] și Esau-Williams [E-3].

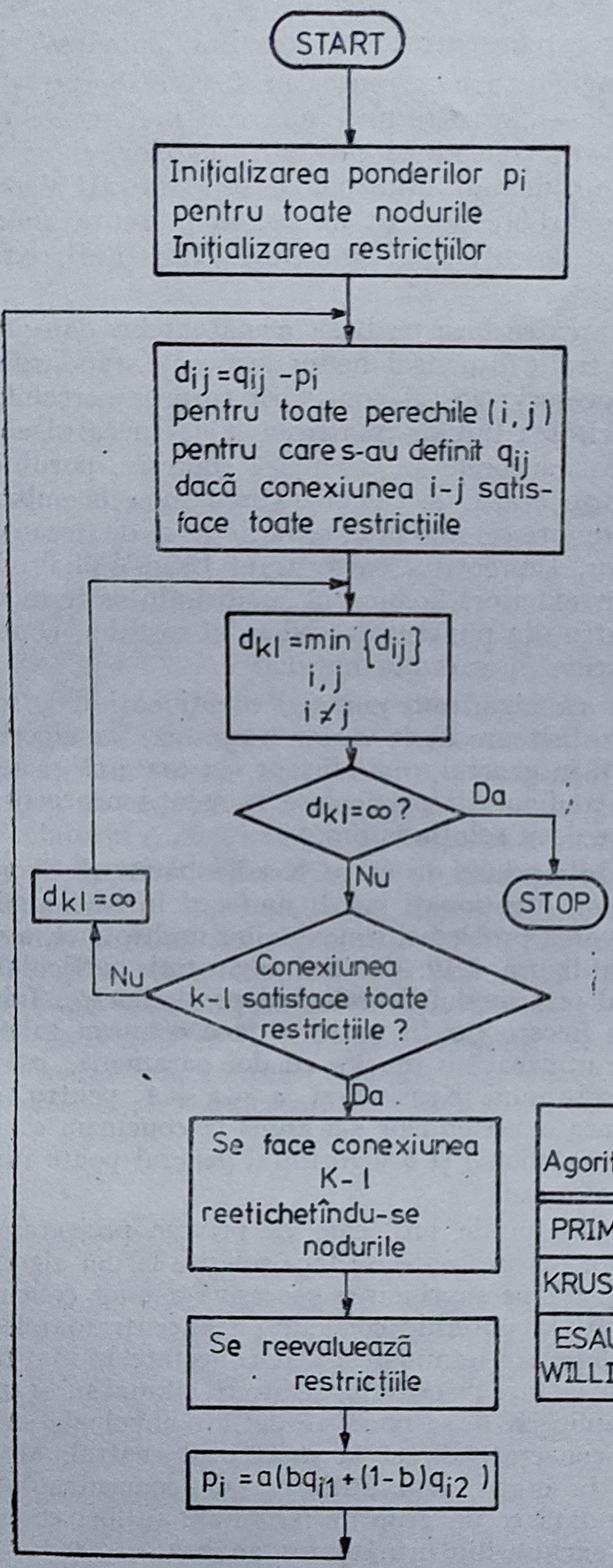
În toate cazurile optimizarea se face pe baza următoarelor date: nodurile, caracterizate de cerința de trafic (numărul mediu de mesaje standard emise în unitatea de timp către procesor); costul conectării tuturor perechilor de noduri (matricea (Q)) și capacitatea liniilor (matricea (C)). În cazul algoritmului euristic elaborat de Prim arborele se generează succesiv, pornindu-se de la calculatorul central, într-o etapă de calcul adăugîndu-se la subarbore nodul al cărui cost de conectare este cel mai mic, urmărindu-se de fiecare dată respectarea tuturor restricțiilor. Dimpotrivă, algoritmul Esau-Williams construiește arborele pornind din exterior, la început analizîndu-se terminalele cele mai „îndepărtate" de centru din punctul de vedere al costului, conectîndu-se succesiv la nodurile vecine în varianta optimă.

Diverse studii au arătat că algoritmii euristici menționați dau soluții în medie cu 5%—15% mai costisitoare decît varianta optimă, iar algoritmul Esau-Williams s-a dovedit a fi în general, mai eficient, cu toate că și acesta, poate conduce pentru unele configurații particulare la soluții neacceptabile (chiar cu 30% mai costisitoare decît soluția optimă).

În urma investigațiilor întreprinse de către Kershenbaum și Chou s-a ajuns la concluzia că algoritmii menționați pot fi unificați într-un algoritm euristic general pentru rezolvarea problemei conexiunilor multipunct, a cărui diagramă este prezentată în figura 7.19. Cei trei algoritmi particulari se regăsesc în algoritmul general prin modul de calcul al ponderilor p_i. Inițializarea acestor ponderi pentru fiecare caz în parte se face conform tabelului 7.5, iar la reevaluarea lor se utilizează o funcție cu doi parametri, $a \geqslant 0$ și $0 \leqslant b \leqslant 1$ ($a = 0$ în cazul algoritmului Kruskal și $a = b = 1$ pentru algoritmul Esau-Williams). Pe baza experiențelor s-a ajuns la concluzia că printr-o alegere judicioasă a parametrilor a și b algoritmul general poate furniza soluții foarte apropiate de cea optimă.

Pentru rezolvarea ansamblului de probleme ce privesc proiectarea și optimizarea topologiei rețelelor arborescente, se face referire la doi algoritmi. Algoritmul [F-2], care se bazează pe suprimarea succesivă a unor concentratoare dintr-o configurație inițială puternic conectată (concentratoarele sînt plasate în toate locurile posibile, iar terminalele sînt conectate la mai multe concentratoare); eliminarea se va face pe criteriul reducerii cît mai substanțiale a costului. În cazul algoritmului [K-6] se pornește de la o topologie stea, în care toate terminalele sînt conectate direct la procesorul central, al cărui cost urmează să fie redus, în etape, prin utilizarea de concentratoare în locurile cele mai bune. De îndată ce un grup de terminale au fost conectate la un concentrator, ele sînt excluse din următoarea analiză.

În ultimul timp se observă tendința unor firme străine de a pune la dispoziția proiectanților pachete de programe de optimizare a topologiei STI,



Tabelul 7.5

Agoritmul	Inițializarea ponderilor p_i
PRIM	$p_1 = 0, p_i = -\infty, i = \overline{2,n}$
KRUSKAL	$p_i = 0, i = \overline{1,n}$
ESAU-WILLIAMS	$p_i = q_{i1}, i = \overline{1,n}$

Fig. 7.19. **Algoritmul unificat pentru realizarea conexiunilor multipunct**

ce urmează să fie deservite de rețeaua de telecomunicație pe care o gestionează. Un asemenea exemplu îl constituie pachetul de programe ORTI [G-3].

- **Optimizarea topologiei rețelelor plasă**

În acest caz localizarea surselor de date, precum și a punctelor de joncțiune a traficului este, de regulă, cunoscută. Sarcina proiectantului este de a realiza interconectarea nodurilor astfel încît costul să fie minim, respectîndu-se cerințele pentru timpul mediu de traversare, capacitatea de transport și siguranță în funcționare. Încărcarea liniilor trebuie să se încadreze într-un anumit domeniu de valori: limita inferioară a domeniului se datorează utilizării eficiente a liniilor, iar cea superioară este dată de capacitatea maximă a canalelor.

Criteriul de fiabilitate poate fi formulat printr-o limită inferioară a conectivității grafului, în cele mai multe cazuri structurile de conectivitate doi — care asigură cel puțin un drum suplimentar dintre oricare pereche de noduri — fiind admisibile. Și în acest caz algoritmii cei mai utilizați sînt euristici, bazați pe modificarea unei structuri inițiale. Existența unei configurații de start face ca soluția rezultată în urma modificărilor să fie dependentă de aceasta. Vor fi necesare cît mai multe asemenea soluții „locale", pentru ca prin compararea lor să se obțină o structură cît mai bună. S-a dovedit că eficiența acestor algoritmi depinde hotărîtor de criteriile pe baza cărora se operează adăugările, ștergerile sau modificările unor linii. Un asemenea algoritm este prezentat în [S-9], dar practica a demonstrat că aceste metode necesită un mare număr de iterații și, implicit, un timp de calcul considerabil. Mai eficienți s-au dovedit a fi algoritmii bazați pe „saturarea tăieturilor". Prezentăm pe scurt această metodă și un algoritm exemplificativ [S-2]. Modificările urmează să fie operate iterativ pe o structură inițială, care se introduce împreună cu datele de intrare: limita inferioară (Rmin) și superioară (Rmax) pentru fluxurile pe linii, numărul maxim de iterații (Imax), conectivitatea ș.a. Valorile Rmin și Rmax pot fi precizate fie sub formă scalară — cînd această limită este unică pentru toate liniile —, fie ca matrice. Repartiția unei sarcini de trafic în rețea, în condițiile respectării cerinței pentru timpul mediu de traversare, este realizată de către procedura ROUTE. Acceptarea sau respingerea unei configurații din punctul de vedere al criteriului de fiabilitate se face prin testarea conectivității.

Metoda se bazează pe depistarea și înlăturarea succesivă a „slăbiciunilor" structurii, considerînd slăbiciune pentru o configurație atît faptul că ea nu răspunde cerințelor de transport datorită „strangulării" traficului pe anumite direcții, cît și utilizarea ineficientă (sub limita Rmin) a unor linii. Remediul în prima situație constă în adăugarea de linii suplimentare, care să preia surplusul de trafic, iar în al doilea caz se procedează la înlăturarea unor linii încărcate ineficient. Evident, ambele operațiuni implică o redistribuire a traficului în rețea, așa încît procesul va trebui repetat. Chiar dacă o topologie nu prezintă slăbiciuni în sensul menționat, ea urmează să fie supusă transformărilor în scopul optimizării.

Algoritmul utilizează următoarele proceduri:

1. ROUTE, distribuie optim o anumită cerință de trafic pe linii, determinîndu-se elementele matricei (R), urmărindu-se minimizarea timpului mediu de traversare.

2. CUTSET&ADD, realizează depistarea tăieturii saturate și adăugarea unei linii suplimentare, cu rol de „punte" peste tăietura respectivă. Descoperirea tăieturii critice se face prin mărirea treptată a cerinței de transport în rețea, de fiecare dată fluxul urmînd a fi dirijat prin apelarea procedurii ROUTE, pînă cînd apare o tăietură de linii saturată într-o anumită proporție. La setul respectiv de linii urmează să fie adăugată o nouă legătură, în așa fel ca să nu se afecteze substanțial costul. Pentru aceasta se invocă criteriul „distanța 2" [D-10].

3. DELETE, realizează eliminarea unei linii dintr-o configurație puternic conectată și deci costisitoare. Legătura suprimată va fi cea mai ineficient utilizată și costisitoare, pentru care expresia $q_i(C_i - f_i)/C_i$ este maximă (q_i reprezintă costul liniei, C_i capacitatea sa, iar f_i fluxul pe care îl transportă).

4. BACK, reface configurația anterioară unor modificări, care s-au dovedit a fi ineficiente în sensul satisfacerii cerințelor.

Structura algoritmului este prezentată în figura 7.20.

Prin utilizarea algoritmului descris se obține o configurație de cost scăzut, care satisface cerințele de performanță impuse, și prezintă o bună stabilitate (apariția fenomenului de congestie în rețele se datorează și saturării unor tăieturi).

7.4.2. OPTIMIZAREA CAPACITĂȚII ȘI A FLUXULUI PE LINII

Optimizarea transportului datelor într-o rețea cu topologia fixată implică rezolvarea următoarelor probleme: ce capacitate trebuie să fie disponibilă pe fiecare linie pentru a asigura o anumită cerință de transport, astfel încît timpul de răspuns să fie minim; apoi — odată capacitățile liniilor stabilite —, cum urmează să fie repartizat fluxul informațional pentru a se utiliza cît mai eficient capacitatea globală a rețelei.

Problema capacității și a fluxului este unitară, doar pentru abordarea ei s-au formulat cele două subprobleme. Chiar și etapizată ea rămîne complexă, din care cauză se vor face unele ipoteze simplificatoare. Canalele liniei se presupun de capacități egale, deși, în realitate, fluxurile în cele două sensuri pot să difere substanțial. Această condiție, valabilă pentru toate liniile rețelei, va simetriza matricea (R), permițînd analiza traficului doar într-o singură direcție. De asemenea, vor fi neglijați traficul pasiv, întîrzierile datorate prelucrărilor în noduri precum și timpii de propagare pe linii. Sursele de date sînt considerate de tip Poisson, iar lungimea tuturor mesajelor va fi presupusă unică, de $1/\mu$ biți. În aceste condiții, conform celor arătate în paragraful 7.3.1, din relația 7.46 se obține media timpului de traversare a nodului i:

$$T_i = \frac{1}{\mu C_i - \lambda_i} \cdot \tag{7.78}$$

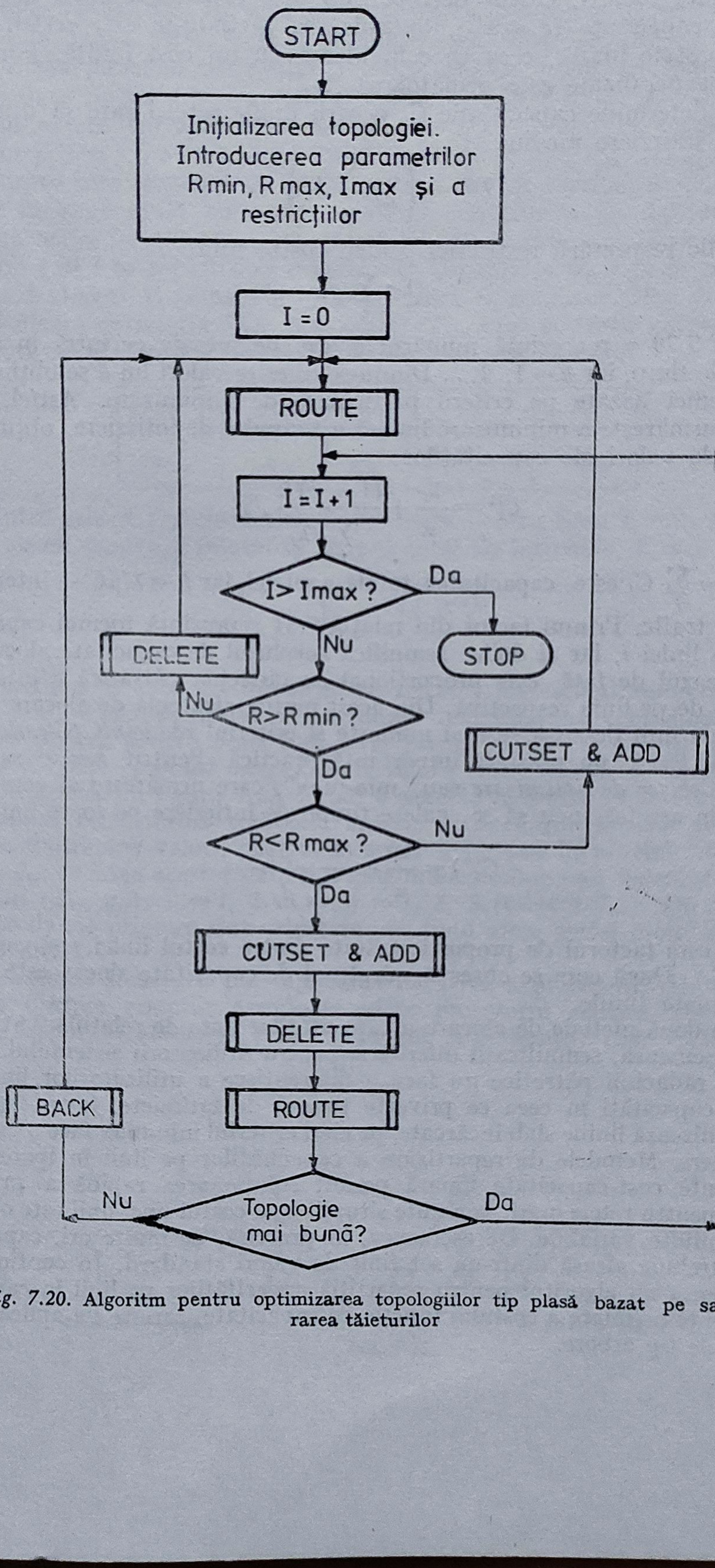


Fig. 7.20. Algoritm pentru optimizarea topologiilor tip plasă bazat pe saturarea tăieturilor

În multe lucrări, costul fiecărei linii este considerat ca o dependență liniară de capacitate ($q = dC$), optimizarea făcîndu-se sub restricția unei capacități totale limită, ceea ce echivalează cu un cost limită. Formularea problemei generalizate este următoarea [S-2]:

Să se determine capacitățile C_i pentru liniile rețelei care să minimizeze timpul de întîrziere mediu

$$T^{(k)} = \left(\sum_i \frac{\lambda_i}{\gamma} T_i^k \right)^{1/k}, \tag{7.79}$$

în condițiile respectării restricției pentru costul total

$$Q \leqslant \sum_i q_i. \tag{7.80}$$

În relația 7.79 γ reprezintă numărul mediu de mesaje ce intră în rețea în unitatea de timp, iar $k = 1, 2, \ldots$ Dîndu-se diferite valori lui k se obțin soluții ale problemei bazate pe criterii particulare de minimizare. Astfel, pentru $k = 1$ se urmărește o minimizare liniară a timpului de întîrziere, obținîndu-se următoarele valori ale capacităților:

$$C_i^{(1)} = \frac{\lambda_i}{\mu} + \frac{(1 - \rho) C}{\sum_j \sqrt{\lambda_j}} \sqrt{\lambda_i}, \tag{7.81}$$

unde $C = \sum_j C_j$ este capacitatea totală a rețelei, iar $\rho = \lambda / \mu C$ — intensitatea totală de trafic. Primul factor din relația 7.81 reprezintă tocmai capacitatea minimă a liniei i, iar al doilea semnifică surplusul de capacitate alocat liniei care, în cazul de față, este proporțional cu rădăcina pătratică a intensității traficului de pe linia respectivă. Din acest motiv, strategia de alocare a capacităților pe linii descrisă se mai numește și criteriul *rădăcinii pătratice.*

Cazul $k = 2$ nu prezintă importanță practică. Pentru $k = \infty$ se obține *criteriul Cebișev de optimizare* sau „min-max", care urmărește alegerea capacităților în așa fel, încît să se egaleze timpii de întîrziere pe toate liniile:

$$C_i^{(\infty)} = \frac{\lambda_i}{\mu} + \frac{(1 - \rho) C}{\sum_j d_j}, \tag{7.82}$$

unde d_j este factorul de proporționalitate dintre costul liniei j și capacitate ($q_j = d_j C_j$). După cum se observă, surplusul de capacitate alocat este același pentru toate liniile.

Cele două metode de alocare a capacităților date de relațiile 7.81 și 7.82 au, de asemenea, semnificații diferite și pentru utilizatorii sistemului. Astfel, criteriul rădăcinii pătratice nu face o diferențiere a utilizatorilor liniilor cu diferite capacități în ceea ce privește timpul de întîrziere, favorizîndu-i pe cei ce utilizează liniile slab încărcate, pe cînd criteriul min-max face o asemenea diferențiere. Metodele de repartizare a capacităților pe linii în ipoteza unei dependențe cost-capacitate liniară permit soluționarea rapidă a problemei chiar și pentru rețele mari. În multe situații însă costul unei linii este o funcție de mai multe variabile. De asemenea, în practică, de multe ori, capacitatea liniilor trebuie aleasă dintr-un set finit de valori standard. În continuare se va prezenta un algoritm pentru repartiția capacităților pe linii în cazul unei dependențe neliniare a costului funcție de capacitate, care se va aplica pentru o rețea de tip arbore.

Metoda [F-3] [S-2] își propune rezolvarea problemei pentru o anumită configurație fixată, astfel încît să se minimizeze timpul mediu de transmitere dintre oricare punct al sistemului și procesorul de prelucrare, pe baza următoarelor date inițiale: n — numărul de perechi nod-linie; λ_i — intensitatea traficului extern corespunzător nodului i; $1/\mu$ — lungimea mesajelor vehiculate în sistem; l_i — lungimea fizică a liniei i; $\{C_1, C_2, ..., C_p\}$ — setul de valori dintre care urmează să se aleagă capacitățile liniilor; d_j — costul unei linii de lungime egală cu unitatea avînd capacitatea C_j. Teoretic, pentru problema astfel formulată există p^n soluții, dar o primă condiție este ca fiecărei linii i să i se repartizeze o asemenea capacitate C_i care să suporte traficul total aferent λ_i' (intern și extern), adică $C_i \geqslant \lambda_i'/\mu$. Pentru fiecare linie i se calculează cu relația 7.78 întîrzierile T_{ij}, apoi costurile corespunzătoare q_{ij}, valori ce se introduc într-un tabel de forma următoare:

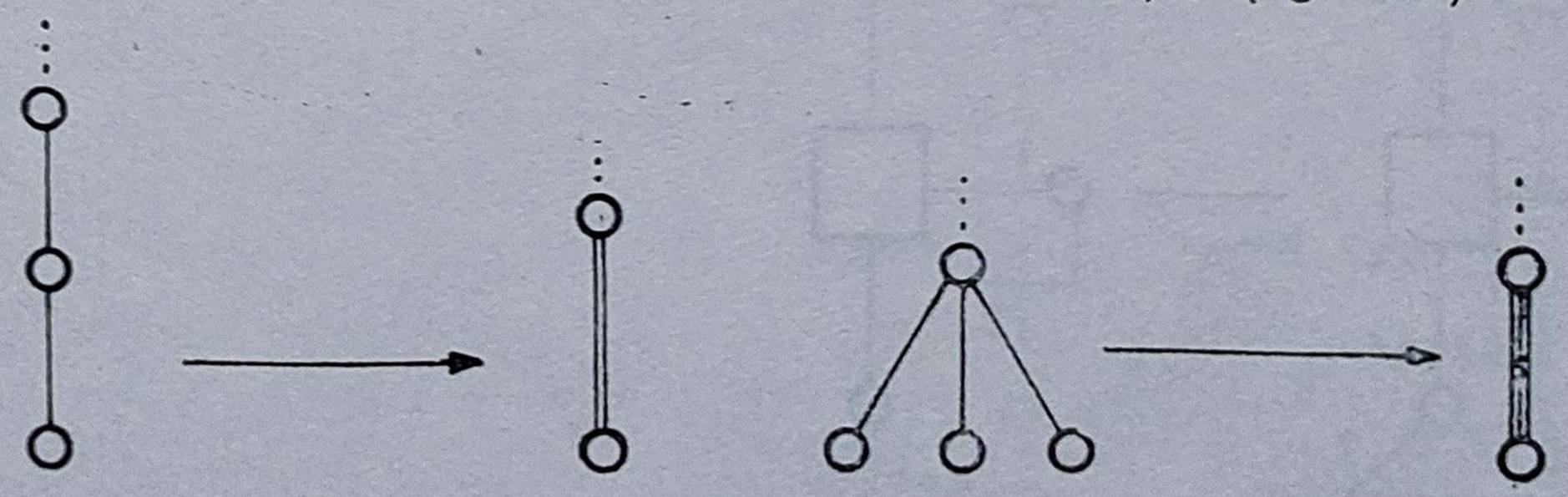
T_{i1}	T_{i2}	T_{i3}	...	T_{ip}
q_{i1}	q_{i2}	q_{i3}	...	q_{ip}

Dintre cele p perechi astfel determinate pentru linia i, una singură va trebui aleasă pentru a minimiza timpul total de întîrziere T, sub restricția:

$$\sum_{\substack{i=1 \\ j \in \{1, 2, ..., p\}}}^{n} q_{ij} < Q.$$

Pentru a se micșora numărul de iterații, algoritmul reduce structura rețelei prin sinteze succesive a mai multe linii într-una echivalentă, distingîndu-se sinteza pentru două linii serie *(serial merge)* și pentru un fascicul de linii paralele *(parallel merge)*, ca în figura 7.21. Metoda se va exemplifica în cazul rețelei din figura 7.22, avînd 8 perechi nod-linie pentru fiecare nod indicîndu-se intensitatea traficului extern. Capacitățile liniilor urmează să fie alese dintre trei valori, pentru fiecare indicîndu-se în tabelul 7.6 costurile unitare d_j. Pe baza acestor date și a lungimilor liniilor s-au calculat perechile de valori (T_{ij}, q_{ij}); $i = 1, 2, ..., 8$; $j = 1, 2, 3$ (tabelul 7.7). De menționat că toate datele primare sînt arbitrare, ele fiind alese astfel încît să simplifice calculele.

Reducerea structurii prin aplicarea iterativă a procedurii de sintetizare se face dinspre nodurile terminale către procesorul central. De exemplu, sinteza liniilor paralele 6 și 7 se efectuează în 5 iterații, (fig. 7.23) conform



a. serial merge *b. paralel merge*

Fig. 7.21.

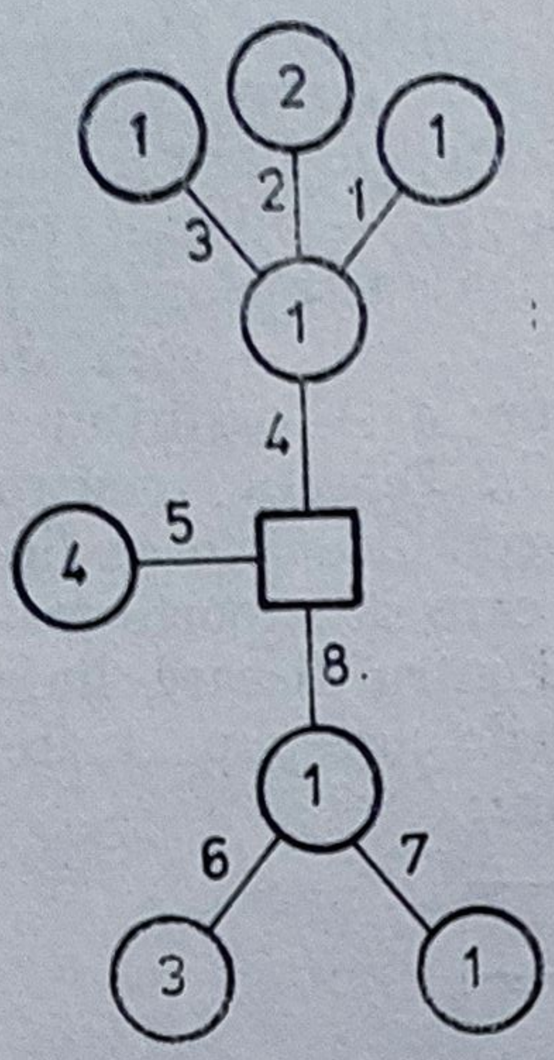


Fig. 7.22. Rețeaua arbore exemplificativă

TABELUL 7.6

Capacitățile C_j [biți/s]	Costurile d_j [u. monetară/km]
720	2
960	3
1 200	4

TABELUL 7.7

linia	lungimea [km]	$C_1 = 720$ *bps*		$C_2 = 960$ *bps*		$C_3 = 1\,200$ *bps*	
		q	T [ms]	q	T [ms]	q	T [ms]
1	10	20	200	30	143	40	111
2	20	40	250	60	166	80	125
3	30	60	200	90	143	120	111
4	40	80	1 000	120	333	160	200
5	50	100	500	150	250	200	166
6	60	120	333	180	200	240	143
7	70	140	200	210	143	280	111
8	80	160	1 000	240	333	320	200

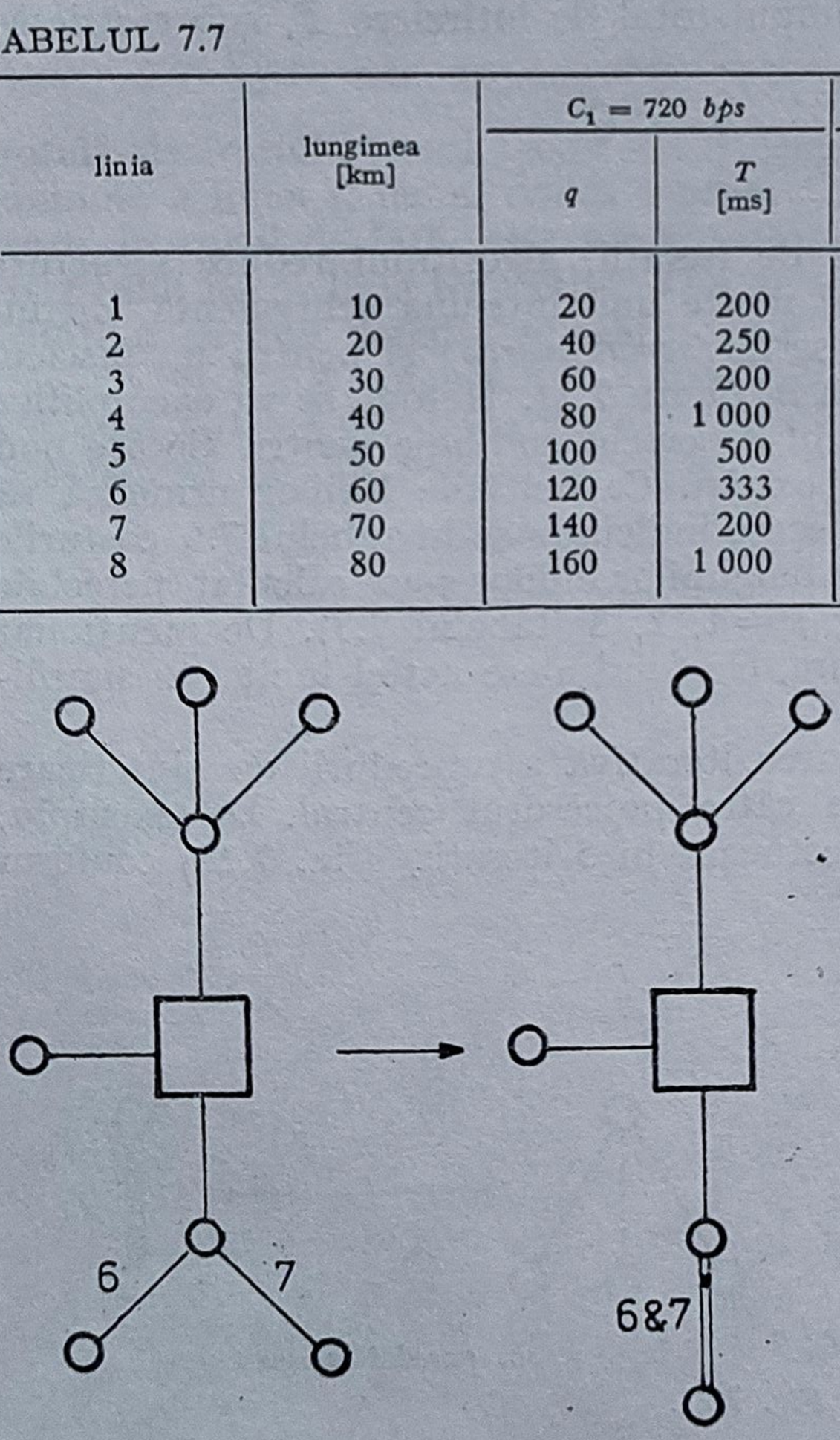


Fig. 7.23. Sinteza liniei echivalente 6&7

tabelului 7.8., iar rezultatul oprației este redat în tabelul 7.9. La început se consideră pentru fiecare linie întîrzierile maxime, după care cea mai mare valoare dintre T_6 și T_7 (valorile cu cifre cursive în tabel) se înlocuiește cu următoarea valoare ca mărime pentru linia respectivă ș.a.m.d. Pentru linia echivalentă, perechile (T, q) se determină cu relațiile $T = \max\{T_6, T_7\}$ și $q = q_6 + q_7$. Dintre cele 5 perechi astfel obținute, pentru linia echivalentă se reține numai 3, întrucît varianta rezultată la iterația a 3-a este mai dezavantajoasă din punctul de vedere al costului decît cea determinată la iterația a 2-a, întîrzierile fiind aceleași. Pe același considerent a fost eliminată și combinația obținută la iterația a 5-a.

TABELUL 7.8

Linia	6	7	Linia echivalentă
T q indici C_j	*333* 120 1	200 140 1	333 120 + 140 1_6 1_7
T q indici C_j	200 180 2	*200* 140 1	200 180 + 140 2_6 1_7
T q indici C_j	*200* 180 2	143 210 2	200 180 + 210 2_6 2_7
T q indici C_j	143 240 3	*143* 210 2	143 240 + 210 3_6 2_7
T q indici C_j	143 240 3	111 280 3	143 240 + 280 3_6 3_7

TABELUL 7.9

T q	333 260	200 320	143 450
	$1_6 1_7$	$2_6 1_7$	$3_6 2_7$

Sinteza a două linii serie se va exemplifica pentru combinarea liniei echivalente 6&7, determinată anterior, cu linia 8 (fig. 7.24). Procedeul este analog, cu deosebirea că întîrzierile pentru linia echivalentă se obțin prin

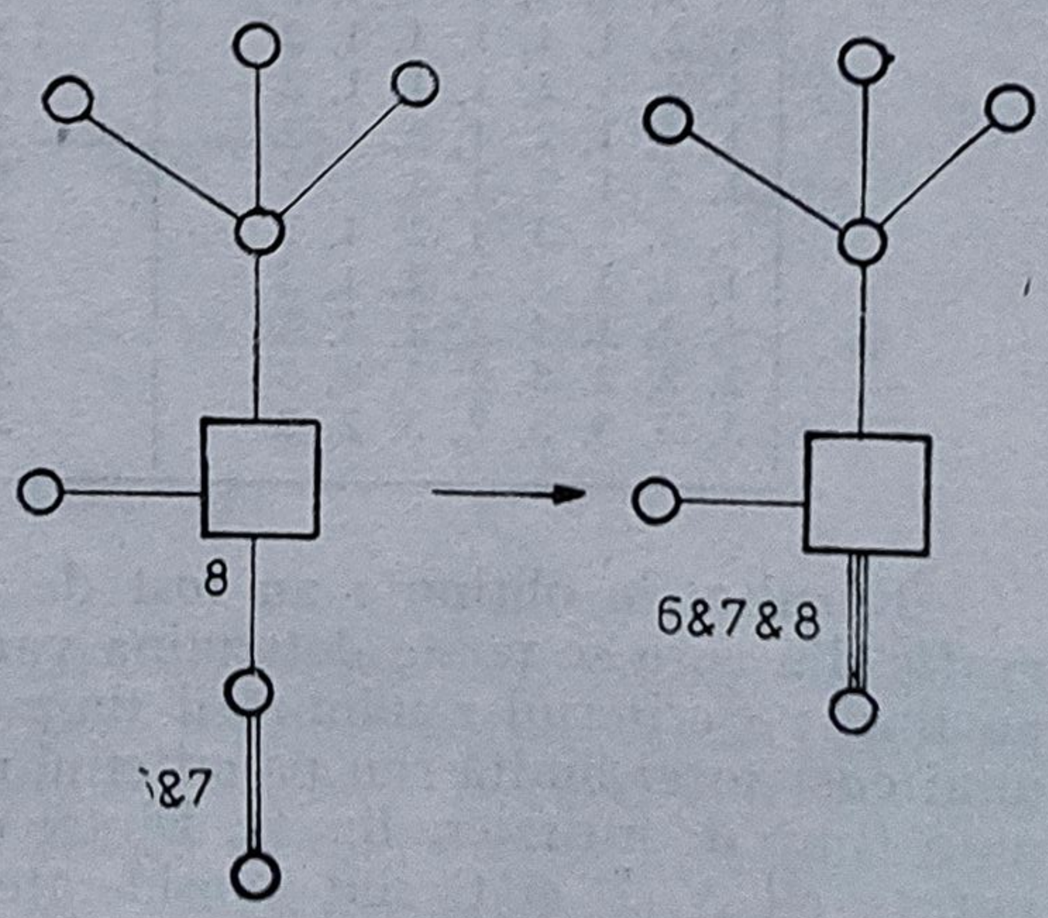


Fig. 7.24. Sinteza liniei echivalente 6&7&8

cumularea întîrzierilor de pe cele două linii $T = T_{6\&7} + T_8$. Iterațiile sintezei sînt prezentate în tabelul 7.10, iar caracteristicile liniei echivalente 6&7&8 se pot urmări în tabelul 7.11.

TABELUL 7.10

	6&7	8	Linia echivalentă
T	333	*1 000*	1 000 + 333
q	260	160	260 + 160
	1,1	1	1, 1, 1
T	*333*	333	333 + 333
q	260	240	260 + 240
	1,1	2	1, 1, 2
T	200	*333*	200 + 333
q	320	240	320 + 240
	2,1	2	2, 1, 2
T	*200*	200	200 + 200
q	320	320	320 + 320
	2,1	3	2, 1, 3
T	143	200	143 + 200
q	450	320	450 + 320
	3,2	3	3, 2, 3

TABELUL 7.11

T	1 333	666	533	400	343
q	420	500	560	640	770
	1, 1, 1	1, 1, 2	2, 1, 2	2, 1, 3	3, 2, 3

Prin aplicarea procedurii de sinteză încă de trei ori (sinteza paralelă a liniilor 1, 2 și 3, apoi cea serială pentru liniile 1&2&3 și 4 și, în final, sinteza liniilor paralele 1&2&3&4, 5 și 6&7&8) se obțin pentru întreaga rețea datele din tabelul 7.12.

TABELUL 7.12

Indicii capacităţilor liniilor	T [ms]	Q_{total}	Varianta
1, 1, 1, 1, 1, 1, 1, 1	1 333	720	
1, 1, 1, 1, 1, 1, 1, 2	1 250	800	
1, 2, 1, 1, 1, 1, 1, 2	1 200	820	
1, 1, 1, 2, 1, 1, 1, 2	666	840	A
1, 1, 1, 2, 1, 2, 1, 2	583	900	
1, 2, 1, 2, 1, 2, 1, 2	533	920	B
1, 1, 1, 3, 1, 2, 1, 3	500	1 020	
1, 1, 1, 3, 2, 2, 1, 3	450	1 070	
1, 2, 1, 3, 2, 2, 1, 3	400	1 090	C
2, 2, 2, 3, 2, 3, 2, 3	366	1 260	
3, 3, 3, 3, 2, 3, 2, 3	343	1 320	

Rezultatele obținute au fost de asemenea reprezentate în figura 7.25, grafic din care se poate determina varianta optimă de alocare a capacităților pe linii pe criteriul minimizării timpului maxim de întîrziere sub restricția unui cost total limită sau pe criteriul minimizării costului total sub restricția unui timp de întîrziere limită. Dintre cele 11 combinații rezultate, variantele notate cu A, B și C sînt considerate mai reprezentative.

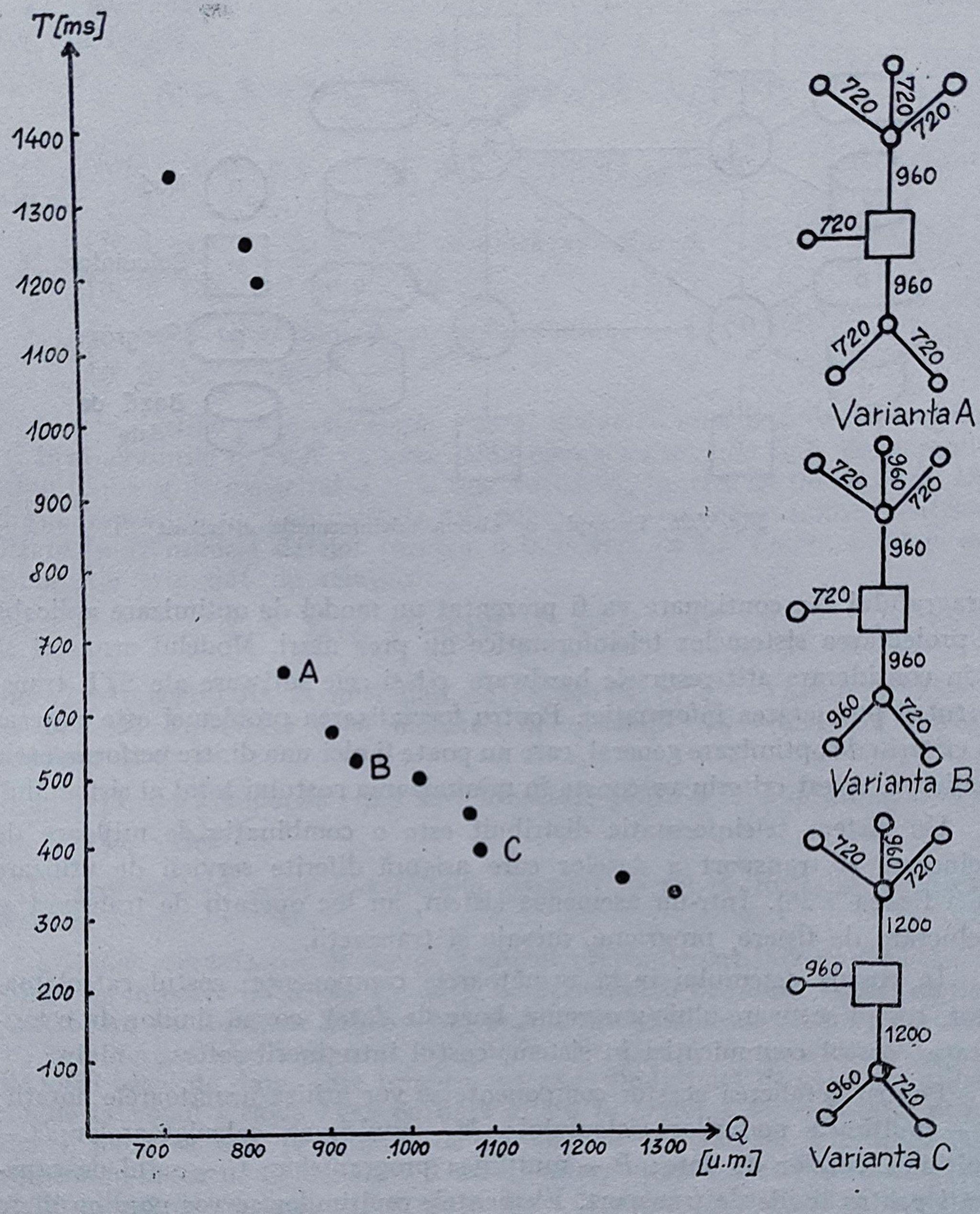


Fig. 7.25. Variantele de alocare a capacităților pe liniile rețelei din figura 7.22

7.4.3. COSTUL STI. OPTIMIZAREA STI ÎN FUNCȚIE DE COST

Datorită complexității optimizării STI problema generală a fost etapizată, ocupîndu-ne pînă în prezent de cîteva dintre aspectele ei. O asemenea abordare este utilă, deoarece — după cum s-a arătat — în practica proiectării pot interveni restricții simplificatoare, situații în care optimizarea se reduce la rezolvarea uneia sau unora dintre subproblemele enunțate la începutul

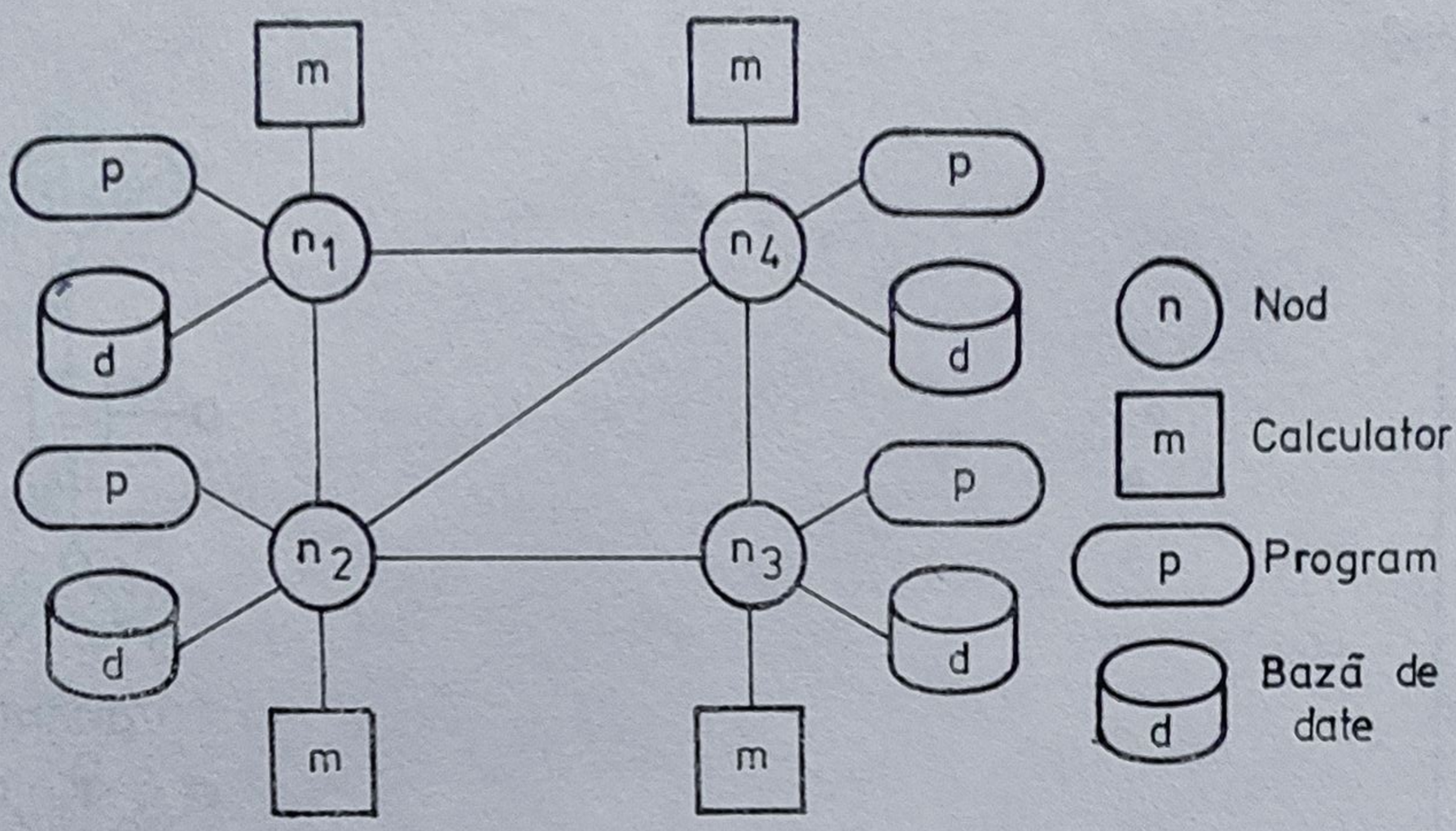


Fig. 7.26. Exemplu de sistem teleinformatic distribuit

paragrafului. În continuare va fi prezentat un model de optimizare aplicabil în proiectarea sistemelor teleinformatice nu prea mari. Modelul urmează să ia în considerare atît resursele hardware, cît și cele software ale STI, transportul și prelucrarea informației. Pentru formalizarea problemei este necesar un criteriu de optimizare general, care nu poate fi nici una dintre performanțele analizate. Acest criteriu va consta în minimizarea costului total al sistemului.

Un sistem teleinformatic distribuit este o combinație de mijloace de prelucrare și transport a datelor care asigură diferite servicii de utilizare (vezi figura 7.26). Într-un asemenea sistem, au loc operații de transport și prelucrare de fișiere, programe, mesaje și tranzacții.

În costul sistemului intră următoarele componente: costul calculatoarelor, costul software-ului (programe, baze de date), costul liniilor de comunicație, costul comunicației în sistem, costul întreținerii software-ului.

Pentru detalierea acestor componente se vor utiliza următoarele notații: N — mulțimea nodurilor sistemului; M — mulțimea calculatoarelor; D — mulțimea bazelor de date; P — mulțimea programelor; C — setul de capacități pentru liniile de transport. Elementele mulțimilor se vor nota cu litere mici, iar costurile specifice le vom considera pe unitatea de timp.

• *Costul calculatoarelor* este dat de expresia:

$$Q_1 = \sum_{n,\,m} q_m \delta_n^m, \quad n \in N \quad \text{și} \quad m \in M, \tag{7.83}$$

unde q_m este costul unui calculator de tip m, pe unitatea de timp, iar

$$\delta_n^m = \begin{cases} 1, \text{ dacă calculatorul de tip } m \text{ intră în dotarea nodului } n, \\ 0, \text{ în caz contrar.} \end{cases}$$

• *Costul software-ului* cuprinde costul implementării bazelor de date și a programelor. El se determină cu ajutorul expresiei:

$$Q_2 = \sum_{n,\,d} q_d \delta_n^d + \sum_{n,\,p} q_p \delta_n^p, \quad n \in N,\ d \in D,\ p \in P, \tag{7.84}$$

în care q_d este costul unei baze de date de tip d; q_p — costul unui program de tip p;

$$\delta_n^d = \begin{cases} 1, \text{ dacă baza de date } d \text{ se alocă nodului } n, \\ 0, \text{ în caz contrar;} \end{cases}$$

$$\delta_n^p = \begin{cases} 1, \text{ dacă programul } p \text{ se alocă nodului } n, \\ 0, \text{ în caz contrar.} \end{cases}$$

• *Costul liniilor de comunicație.* Costul instalării unei linii de capacitate $c \in C$ între nodurile $i,\ j \in N$ va avea două componente; una fixă, ce depinde de tipul liniei și de capacitatea c, și una variabilă, funcție de distanța dintre nodurile respective, l_{ij}. Componenta fixă poate include și costul echipamentelor utilizate în transmisia datelor (pentru detalii vezi [S-2]). Costul liniilor de comunicație este dat de relația:

$$Q_3 = \sum_{i,\,j,\,c} (q'_c + l_{ij} q''_c)\, \delta_{ij}^c, \quad i,\ j \in N,\ c \in C, \tag{7.85}$$

unde q'_c este costul fix al unei linii de capacitate c; q''_c — costul instalării unei linii de capacitate c pe unitatea de distanță; l_{ij} — distanța dintre nodurile i și j;

$$\delta_{ij}^c = \begin{cases} 1, \text{ dacă nodurile } i \text{ și } j \text{ sînt conectate printr-o linie de capacitate } c \\ 0, \text{ altfel.} \end{cases}$$

De remarcat că acest cost trebuie deosebit de cel de utilizare efectivă a liniilor.

• *Costul comunicației în sistem.* Traficul preponderent într-un STI este destinat fie interogării bazelor de date, fie actualizării acestora. O tranzacție de interogare sau o punere la zi se face prin intermediul unui program specializat, care se poate afla într-un loc diferit atît de cel al utilizatorului, cît și de locul bazei de date. Prin urmare, o tranzacție inițiată de către un utilizator de la nodul i către baza de date aflată la nodul k va fi rutată prin nodul j, unde se află programul de actualizare, p, traficul informațional desfășurîndu-se ca în figura 7.27. Traficul informațional dintre utilizator și program diferă de cel dintre program și baza de date. De asemenea, trebuie

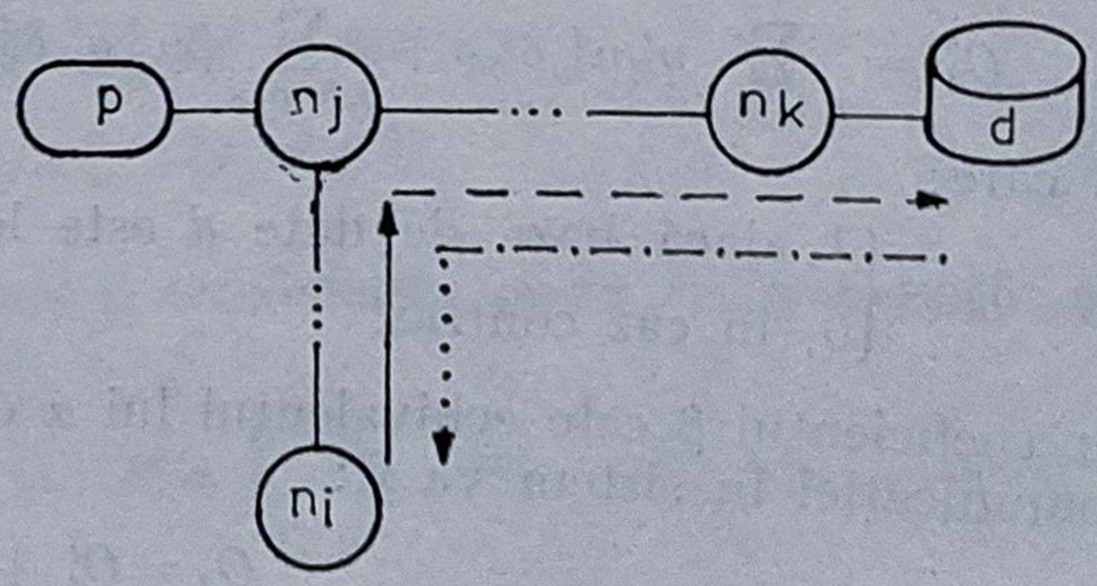


Fig. 7.27. Prelucrarea unei tranzacții într-un STI distribuit

să avem în vedere și informația retur, de răspuns, chiar dacă în calcule, pentru simplificare, se neglijează.

— *Costul traficului de interogare* dintre utilizator și programul p va fi caracterizat de mărimile:

λ_{ip}^{d} — numărul de tranzacții în unitatea de timp dintre nodul i către baza de date d, via programul p din nodul j;

q'_{ij} — costul de comunicație a unei tranzacții pe linia (i, j);

$$\delta_{ijd}^{p} = \begin{cases} 1, \text{ dacă o tranzacție de la nodul } i \text{ către baza de date } d \text{ este} \\ \text{rutată către programul } p \text{ din nodul } j; \\ 0, \text{ în caz contrar} \end{cases}$$

γ'_{ij} — procentul estimat pentru traficul de răspuns la interogările λ_{ip}^{d}.

Traficul total de interogare a bazei de date d prelucrat de către programul p din nodul j va fi:

$$\lambda_{jp}^{d} = \sum_{i} \lambda_{ip}^{d} \delta_{ijd}^{p}.$$

Pentru determinarea traficului dintre nodul j și baza de date d, se notează:

$$\delta_{jkp}^{d} = \begin{cases} 1, \text{ dacă tranzacția prelucrată de către programul } p \text{ din nodul } j \\ \text{vizează baza de date } d \text{ din nodul } k, \\ 0, \text{ în caz contrar} \end{cases}$$

α — raportul lungimilor mesajelor de interogare dintre $p \to d$ și $i \to p$.

Cu aceste notații, costul total al traficului din sistem este dat de expresia:

$$Q'_4 = \sum_{i,\,j,\,p,\,d} q'_{ij} \lambda_{ip}^{d} \delta_{ijd}^{p} (1 + \gamma'_{ij}) + \sum_{j,\,k,\,p,\,d} \alpha q''_{jk} \lambda_{jp}^{d} \delta_{jkp}^{d} (1 + \gamma''_{jk}), \quad i, j, k \in N, \; p \in P, \; d \in D. \tag{7.86}$$

Primul termen al relației 7.86 semnifică costul traficului de interogare dintre utilizatori și programe, iar al doilea, dintre programe și bazele de date (mărimile q''_{jk} și γ''_{jk} au semnificația echivalentă cu cele din primul termen).

— *Costul traficului de actualizare*. Pe baza acelorași considerente, dacă μ_{ip}^{d} este traficul de actualizare dintre nodul i și baza de date d, via programul p, atunci traficul total de actualizare prelucrat de către programul p din nodul j va fi:

$$\mu_{jp}^{d} = \sum_{i} \mu_{ip}^{d} \delta_{ijd}^{p}.$$

Costul total al traficului de actualizare a informației din sistem se determină cu ajutorul expresiei:

$$Q''_4 = \sum_{i,\,j,\,p,\,d} q'_{ij} \mu_{ip}^{d} \delta_{ijd}^{p} + \sum_{j,\,k,\,p,\,d} \beta \mu_{jp}^{d} q''_{jk} \delta_{k}^{d}, \quad i, j, k \in N, \; p \in P, \; d \in D, \tag{7.87}$$

în care:

$$\delta_{k}^{d} = \begin{cases} 1, \text{ dacă baza de date } d \text{ este localizată în nodul } k, \\ 0, \text{ în caz contrar.} \end{cases}$$

iar coeficientul β este echivalentul lui α din cazul precedent. Atunci, costul comunicației în sistem va fi:

$$Q_4 = Q'_4 + Q''_4.$$

• *Costul întreținerii software-ului* cuprinde cheltuielile pentru memorarea și administrarea bazelor de date și a programelor. Fie q_n^d — costul de întreținere a bazei de date d la nodul n, iar q_n^p = costul de întreținere a programului p la nodul n. În acest caz, costul total de întreținere a software-ului este dat de:

$$Q_5 = \sum_{n,\,d} (q_n^d \delta_n^d + q_n^p \delta_n^p), \tag{7.88}$$

unde δ_n^d și δ_n^p reprezintă variabilele binare de apartenență ale bazei de date d respectiv ale programului p, la nodul n, definite la analiza costului de implementare a software-ului.

Modelul de optimizare

În [C-5] este prezentat un model de optimizare care rezolvă multe dintre problemele sistemelor teleinformatice distribuite: alocarea capacităților pe linii, distribuția puterii de calcul, alocarea programelor și a bazelor de date pe noduri ș.a. Modelul prezintă următoarele avantaje:

— nu se pornește de la o anumită topologie de rețea, ci de la utilizatori caracterizați de anumite cerințe de trafic și o anumită răspîndire geografică. Algoritmul urmează să decidă modul de conectare a tuturor punctelor integrate în sistem, capacitatea fiecărei linii și disciplina de control al fluxului informațional;

— alocarea puterii de calcul pe noduri se va face dintre procesoarele disponibile, fiecare tip fiind caracterizat de o capacitate limitată;

— modelul ia în considerare dependența dintre programe și bazele de date: o tranzacție destinată unei anumite baze de date urmează să fie prelucrată de către un anumit program;

— se ia în calcul și fluxul informațional de retur, nu numai traficul inițiat de către utilizatori. Răspunsul este considerat de o anumită pondere din informația directă, dar — pentru simplificare — se va presupune că el urmează aceeași rută.

Funcția obiectiv urmărită în model este minimizarea costului total al sistemului:

$$Q = \sum_{i=1}^{5} Q_i, \tag{7.89}$$

Q_i fiind costurile parțiale determinate anterior.

Pentru ca o soluție să fie acceptabilă, ea va trebui să îndeplinească următoarele condiții (restricțiile modelului):

a. graful corespunzător rețelei să fie conectat:

$$\sum_{j,\,c} \delta_{ij}^c \geqslant 1, \quad (\forall)\ i \in N; \tag{7.90}$$

b. existența resurselor software și coexistența lor cu procesoarele, astfel:

— pentru programe

$$\sum_{n} \delta_n^p \geqslant 1, \quad (\forall)\ p \in P \quad \text{și} \quad \delta_n^p \leqslant \sum_{m} \delta_n^m, \quad (\forall) n \in N,\ p \in P \tag{7.91}$$

— pentru bazele de date

$$\sum_n \delta_n^d \geqslant 1, \quad (\forall)\ d \in D \quad \text{și} \quad \delta_n^d \leqslant \sum_m \delta_n^m, \quad (\forall)\ n \in N,\ d \in D; \tag{7.92}$$

c. cerința de calcul adresată unui nod k de către celelalte noduri nu trebuie să depășească capacitatea procesoarelor din nodul respectiv. Cu notațiile introduse la evaluarea costului de comunicație și a_m semnificînd capacitatea unui procesor de tip m avem:

$$\sum_{i,\,j,\,p,\,d} \alpha \lambda_{ip}^d \delta_{ijd}^p \delta_{jkp}^d + \sum_{i,\,p,\,d} \beta \mu_{ip}^d \delta_k^d + \sum_{i,\,p,\,d} (\lambda_{ip}^p + \mu_{ip}^d)\, \delta_{ikd}^p \leqslant \sum_m \alpha_m \delta_k^m, (\forall)\, k \in N; \tag{7.93}$$

d. fluxul ce urmează să fie dirijat pe o anumită linie, nu trebuie să depășească capacitatea liniei respective. În ipoteza utilizării de linii full-duplex, această restricție pentru linia (i, j) se va formula prin două relații:

— pentru sensul de la i la j:

$$\sum_{l,\,d,\,p} (\alpha \lambda_{lp}^d \delta_{ijp}^d + \beta \mu_{lp}^d \delta_j^d)\, \delta_{lid}^p + \sum_{l,\,d,\,p} \alpha \lambda_{lp}^d \gamma_{ji}'' \delta_{ljd}^p \delta_{jip}^d + \sum_{p,\,d} (\lambda_{ip}^d + \mu_{ip}^d)\, \delta_{ijd}^p +$$

$$+ \sum_{p,\,d} \lambda_{jp}^d \gamma_{ji}' \delta_{jid}^p \leqslant \sum_c a_c \delta_{ij}^c, \quad (\forall),\ i, j \in N,\ i < j; \tag{7.94}$$

— pentru sensul de la j la i:

$$\sum_{l,\,p,\,d} (\alpha \lambda_{lp}^d \delta_{jip}^d + \beta \mu_{lp}^d \delta_i^d)\, \delta_{ljd}^p + \sum_{l,\,d,\,p} \alpha \lambda_{lp}^d \gamma_{ij}'' \delta_{lid}^p \delta_{ijp}^d + \sum_{p,\,d} (\lambda_{ip}^d + \mu_{ip}^d)\, \delta_{jid}^d +$$

$$+ \sum_{p,\,d} \lambda_{ip}^d \gamma_{ij}' \delta_{ijd}^p \leqslant \sum_c a_c \delta_{ij}^c, \quad (\forall)\ i, j \in N,\ i > j, \tag{7.95}$$

unde a_c reprezintă capacitatea liniei c;

e. interdependențele dintre programe și bazele de date:

— pentru tranzacțiile de la nodul i pentru baza de date d, să existe o singură rută către programul p rezident în nodul j, adică:

$$\sum_j \delta_{ijd}^p = 1, \quad (\forall)\ i \in N,\ p \in P,\ d \in D; \tag{7.96}$$

— pentru un program p din nodul j să existe numai cîte o singură rută către bazele de date cu care este în relație, adică:

$$\sum_k \delta_{jkp}^d = \delta_j^p, \tag{7.97}$$

$(\forall)\ j \in N$ la care există $p \in P$ și pentru toate $d \in D$ relaționate cu p;

— rezolvarea tranzacțiilor către bazele de date prin intermediul programelor presupune rezidența acestor componente software în anumite noduri. Dacă notăm cu r numărul de noduri, atunci:

$$\sum_i \delta_{ijd}^p \leqslant r \delta_j^p, \quad (\forall)\ \mathrm{j} \in N,\ p \in P,\ d \in D; \tag{7.98}$$

$$\sum_j \delta_{jkp}^d \leqslant r \delta_k^d, \quad (\forall)\ k \in N \tag{7.99}$$

și pentru toate $p \in P$ relaționate cu $d \in D$.

Modelul definit prin relațiile 7.89—7.99 constituie o problemă de programare neliniară, cu variabilele decizionale δ_k^m, δ_k^d, δ_k^p, δ_{ij}^c, δ_{ijd}^p, δ_{jkp}^d întregi (binare).

Exemplu aplicativ

Pentru rezolvarea modelului a fost dezvoltat un algoritm bazat pe metoda [A-5]. Implementarea s-a făcut în limbajul FORTRAN, iar timpul UC mediu de rulare a programului pe un calculator IBM 370/168 a fost de 7 s.

Aplicația care urmează să fie prezentată constă în proiectarea unui sistem teleinformatic bancar, caracterizat prin următoarele date de intrare:

— 4 noduri;

— 2 tipuri de calculatoare, m_1 și m_2, avînd caracteristicile: costurile lunare $q_{m_1} = 45\,000$ și $q_{m_2} = 35\,000$, iar capacitățile de prelucrare lunare $a_1 = 13 \cdot 10^6$ tranzacții și $a_2 = 7 \cdot 10^6$ tranzacții;

— un singur tip de bază de date și un singur tip de program. Costul bazei de date este $q_d = 10^4$, iar costurile de întreținere sînt $q_n^d = 500$ și $q_n^p = 10$, $(\forall)\ n \in N$;

— conectarea nodurilor urmează să se facă în exclusivitate prin linii de capacitate 9 600 biți/s, la următoarele costuri de instalare: costul fix $q_c' = 200$ și costul variabil $q_c'' = 100$. Distanțele dintre noduri sînt: $l_{12} = 500$ km, $l_{13} = 240$ km, $l_{14} = 260$ km, $l_{23} = 290$ km, $l_{24} = 760$ km și $l_{34} = 450$ km;

— cerințele de trafic din cele patru puncte ale sistemului, cît și costurile vehiculării unei tranzacții între noduri sînt date în tabelul 7.13 (s-a presupus că lungimea unei tranzacții este de 60 caractere, iar $q_{ij}' = q_{ij}''$).

TABELUL 7.13

Nodul sursă	Cerința de trafic (nr. de tranzacții pe lună)		Costul unei tranzacții către nodul destinație			
	λ_{ip}^d	μ_{ip}^d	n_1	n_2	n_3	n_4
n_1	1 500 000	650 000	0	0,015	0,03	0,018
n_2	1 340 000	650 000	0,015	0	0,039	0,054
n_3	1 500 000	650 000	0,030	0,039	0	0,021
n_4	1 500 000	650 000	0,018	0,054	0,021	0

Rezolvarea problemei s-a făcut în următoarele ipoteze:

— mesajele dintre un program și o bază de date cu aceeași lungime cu mesajele corespunzătoare dintre utilizator și program ($\alpha = \beta = 1$).

— informația de retur nu a fost luată în calcul ($\gamma_{ij}' = \gamma_{ij}'' = 0$).

Soluția obținută este următoarea (fig. 7.28): sistemul utilizează trei procesoare alocate nodurilor n_1, n_3 și n_4 (la n_1 procesor de tip m_1, iar la nodurile n_3 și n_4 procesoare de tipul m_2, ceea ce înseamnă că $\delta_{n_1}^{m_1} = 1$, $\delta_{n_3}^{m_2} = 1$ și $\delta_{n_4}^{m_2} = 1$). De asemenea, la fiecare nod cu procesor există cîte o copie a bazei de date și a programului. Conectarea dintre noduri este prezentată în figura 7.28.

Prelucrarea tranzacțiilor de la nodurile n_1, n_3 și n_4 se va face pe procesoarele din nodurile respective, iar cele de la nodul n_2 vor fi rutate către resursele din nodul n_1. Costul total al sistemului, obținut prin minimizarea funcției 7.89, este $Q = 298\,880$.

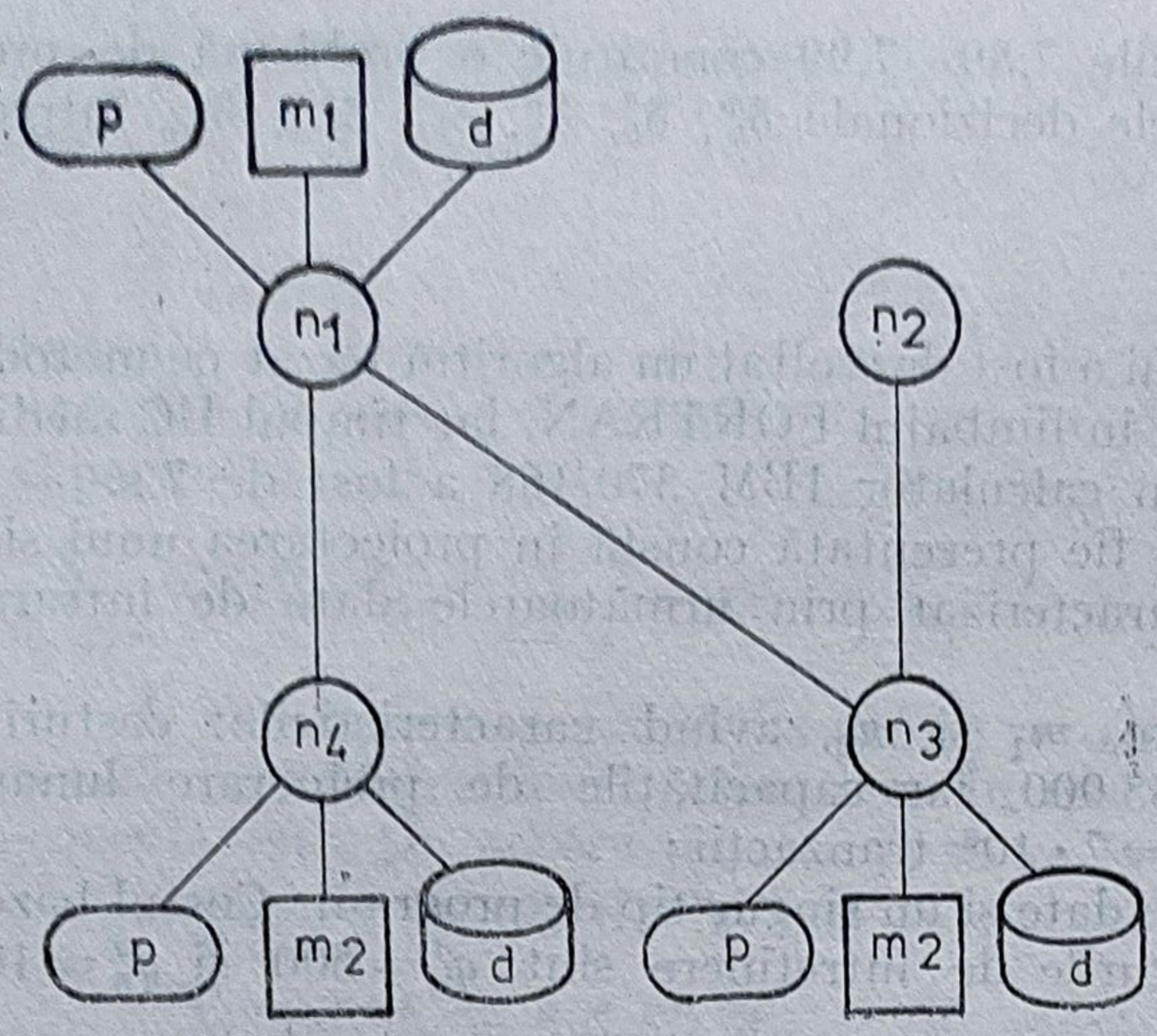


Fig. 7.28. Soluția optimă de proiectare a STI

7.5. CONCLUZII

Proiectarea unui sistem teleinformatic este o sarcină dificilă, considerată încă de pionierat, a cărei regie mai este încă perfectabilă. De asemenea, multe dintre tehnicile utilizate sînt preluate din alte domenii și care urmează să fie îmbunătățite. De fapt, unul dintre scopurile măsurătorilor ce se întreprind asupra STI aflate în funcțiune este și acela de a se obține date care să reflecte comportarea reală a sistemelor, date ce vor servi la îmbunătățirea unor modele utilizate în proiectare. O asemenea preocupare este prezentată în (G-3).

Pentru studiul protocoalelor de comunicație mai nou se utilizează și rețelele Petri. În [M-2] se prezintă un izomorfism între comportarea unei rețele Petri stochastice cu tranziții descrise de legi exponențiale și procesele Markov, tehnică utilizată pentru analiza performanțelor unui protocol orientat pe bit.

În ceea ce privește disponibilitatea unui STI, putem concluziona că aceasta depinde în principal de arhitectura sistemului mai mult sau mai puțin redondantă, de tehnica de semnalizare, localizare și înlăturare a defectelor, precum și de metodele de reconfigurare adoptate. Prin urmare, atenția proiectantului — în dorința de a realiza un sistem cît mai fiabil — va trebui să fie distribuită pentru alegerea componentelor și asigurarea de rezerve pentru cele mai puțin fiabile, interconectarea optimă a subsistemelor, prevederea măsurilor de înlăturare oportună a defectelor și asigurarea unui mod de funcționare, eventual degradat, în situațiile critice. Pentru estimarea performanțelor STI în faza de proiectare, se procedează la analiza parametrilor sistemului prin modelare, simulare sau alte tehnici. Alegerea unui anumit model depinde de tipul rețelei, scopul principal urmărit și exigențele impuse. Cînd în urma analizei se constată că rezultatele nu satisfac cerințele impuse se va acționa asupra unuia sau unui grup de factori ce influențează performanțele, și anume: topologia rețelei de transport; capacitatea și fiabilitatea liniilor; caracteristicile nodurilor (numărul procesoarelor, capacitatea bufferelor etc.); protocoalele de comunicație; tehnica de comandă a fluxului informațional.

BIBLIOGRAFIE SELECTIVĂ

A-1. ANGHELOIU, I.; GYORFI, E.; PATRICIU, V., *Securitatea și protecția informației în sistemele electronice de calcul*, Ed. Militară, București, 1986.

A-2. AGGARWALL, K. K.; GUPTA, J. S.; MISRA, K. B., *A Simple Method for Reliability Evaluation of a Communication System*, IEEE Transactions on Communications, vol. COM-23, may, 1975.

A-3. AGGARWALL, K. K.; GUPTA, J. S.; MISRA, K.B., *A New Method for System Reliability Evaluation*, Microelectronics and Reliability, vol. 12, 1973.

A-4. AGGARWALL, K. K.; CHOPRA, Y. C.; BAJWA, J. S., *Reliability Evaluation by Network Decomposition*, IEEE Transactions on Reliability, vol. R-31, nr. 4, oct., 1982.

A-5. AKOKA, J., *Baunded branch and bound method for mixed inter nonlinear programming*, Sloan Sch. of Management, Massachusetts Inst. Technol., Jan., 1977.

B-1. BROWN, A. P. G., *Progress Towards Distributed Database Systems*, 6th Int. Conf. on Comp. Comm., London, Sept., 1982.

B-2. BOUDIGUE, D., *Bases de données repartis et communications par satellites*, INRIA, no. 103, p. 28—31, sept., 1985.

C-1. CRISTEA, V., *Aspecte privind studiul eficienței instrumentelor de sincronizare a proceselor de calcul paralele*, Buletinul IPB, 1980,3.

C-2. COCULESCU, L.; ANDREI, R.; SIMIONESCU, P.; VASILE, T.; TRIFĂNESCU, D; SOFRONICIU, I.; MARINUȘ, R.; LAMOR, F.; PETROFF, M., *Rețeaua DIGAROM*, ITCI — CM50/86, iunie 1986.

C-3. COCULESCU, L.; MARINUȘ, R.; TRIFĂNESCU, D.; SOFRONICIU, I.; VASILE, T., *Rețeaua DIGAROM — Manualul inginerului de sistem*, ITCI, iunie 1986.

C-4. CĂTUNEANU, V.; BACIVAROF, I., *Fiabilitatea sistemelor de telecomunicații*, Editura Militară, București, 1985.

C-5. CHEN, P. P. S.; AKOKA, J., *Optimal Design of Distributed Information Systems*, IEE Transactions on Computers, vol. C-29, nr. 12, dec. 1980.

C-6. COLE, G. D., *Performance Measurement on the ARPA Computer Network*, IEEE, Transactions on Communications, vol. COM-20, jun., 1972.

D-1. DIJKSTRA, E. W., *Co-operating Sequential processes*, Programming Languages (Genuis, F. ed.), Academic Press, London, 1968.

D-2. DIJKSTRA, E. W., *Guarded commands, nondeterminancy, and formal derivation of programs*, Comm. ACM, 1975, *18*, 8.

D-3. DAVIDOVICIU, A.; BĂRBAT, B., *Limbaje de programare pentru sisteme în timp real* Ed. Tehnică, București, 1986.

D-4. DODESCU, GH.; ODAGESCU, I.; ȘCHEIANU, ȘT.; NĂSTASE, P., *Simularea sistemelor*, Ed. Militară, București, 1986.

D-5. DAVIDA, G. I.; DE MILLO, R. A.; LIPTON, R. J., *Protecting Shared Cryptographic Keys*, Proc. 1980 Symp. on Security and Privacy, IEEE Computer Society, apr., 1980.

D-6. DAVIES, D. W.; PRICE, W. L., *The Application of Digital Signatures Based on Public Key Cryptosystems*, NPL Report DNACS 39/80, dec., 1981.

D-7. DENNING, D. E., *Cryptography and Data Security*, Addison-wesley Publishing Company, 1982.
D-8. DIFFIE, W.; HELLMAN, M., *Privacy and Autentication: An Introduction to Cryptography*, Proc. IEEE, mart. 1975.
D-9. DIFFIE, W.; HELLMAN, M., *New Directions in Cryptography*, IEEE Trans. on Inform. Theory, nov. 1976.
D-10. DAVIES, D. W.; BARBER, D. L. A.; PRICE, W. L.; SOLOMONIDES, C. M., *Computer Networks and Their Protocols*, John Wiley and Sons, 1981.
D-11. DELOBEL, C.; LITWIN, W., *Distributed Data Bases*, N. Holland, INRIA, 1980.

E-1. EHRSAM, W. F.; MATYAS, S. M.; MEYER, C. H.; TUCHMAN, W. L., *A Cryptographic Key Management Scheme for Implementing the DES*, IBM System J., Feb., 1978.
E-2. ECMA-85, *Virtual File Protocol*, sept., 1982.
E-3. ESAU, L. R.; WILLIAMS, K. C., *On teleprocessing system design*, Part II, IBM System Journal, vol. 5, nr. 3, 1966.

F-1. FEISTEL, H.; NOTZ, W. A.; SMITH, J. L., *Some Cryptographic Techniques for Machine to Machine Data Communication*, Proc. IEEE, nov. 1975.
F-2. FELDMAN, E.; LEHNER, F. A.; RAY, T. L., *Warehouse location under continuous economies of scale*, Management Science, vol. 12, 1966.
F-3. FRANK, H.; FRISCH, I. T., *Planning Computer-Communication Network*, Computer-Communication Networks, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1973.

G-1. GELEMBE, E.; PUJOLLE, G., *Introduction aux réseaux de files d'attente*, Ed. Eyrolles, Paris, 1985.
G-2. GELENBE, E.; GRANGE, J. L.; MUSSARD, P., *Performance limits of the TMM protocol: modelling and measurement*, Proceedings of 3-rd Berkeley workshop on distributed systems and computer networks, San Francisco, 1978.
G-3. GRASSIN, J.; GATOT, G., *ORTI, Optimalisation des réseaux de téléinformatique*, Annales des Télécommunications, vol. 28, nr. 1—2, 1973.

H-1. HANSEN, P. B., *Operating System Principles*, Prentice Hall, Inc., Englewood Cliffs, New Jersey, 1973.
H-2. HANSEN, P. B., *Network: A Multiprocessor Program*, IEEE Trans. on Software Engineering, 1978, *SE-4*, 3.
H-3. HOARE, C. A. R., *Monitors: An Operating Systems Structuring Concept*, Comm. ACM, 1974, *17*, 10.
H-4. HOARE, C. A. R., *Communicating Sequential Processes*, Comm. ACM, 1978, *21*, 2.

I-1. * * *, *IBM Systems Journal*, 1987, *26*, 1.
I-2. * * *, *IBM Systems Journal*, 1985, *24*, 2.
I-3. ISO/TC 97/SC 16/N 23, *Proposal for a Standard Virtual Terminal Protocol*, 1982.
I-4. ISO/Tc 97/DIS 8571/1/2/3/4, *File Transfer, Acces and Management*, 1986.

J-1. JASMON, G. B., *Cutset Analysis of Networks Using Basic Minimal Paths and Network Decomposition*, IEEE Transactions on Reliability, vol. R-34, oct., 1985.

K-1. KOBAYASHI, K., *Strategic Approaches to Modern Communications: C & C*, ITU 4-th World Telecommun. Forum, Pt. 1, III. 1, Geneva, oct., 1983.
K-2. KATO, M.; IIMURA, J.; ARAI, K.; HASHIMOTO, S., *Toward Advanced Communications Processing System*, Review of the Electrical Communication Laboratories, 1984, *32*, 6.
K-3. KLEINROCK, L., *Queueing Systems Theory*, vol. I, John Wiley & Sons, N. Y., 1975.
K-4. KUO, F. F., *Protocols and Techniques for Data Communication Network*, Prentice Hall, Englewood Cliffs, N. Y., 1981.
K-5. KRUSKAL, M., *On the shortest spanning subtree of a graph and the travelling salesman problem*, Proc. of American Math. Society, vol. 7, 1956.
K-6. KUEHN, A. A.; HAMBURGER, M. J., *A heuristic program for location warehouses*, Management Science, vol. 9, 1963.

L-1. LAMPSON, B. W. ş.a. ed. *Distributed Systems-Architecture and Implementation*. An Advanced Course, Springer-Verlag, Berlin Heidelberg New York, 1981.

L-2. LAMPORT, L., *A New Solution of Dijkstra's Concurrent Programming Problem*, Comm. ACM, 1974, *17*, 8.

L-3. LAMPORT, L., *Time, Clocks, and Ordering of Events in a Distributed System*, Comm. ACM, 1978, *21*, 7.

L-4. LEUCUȚIA, D.; FOTESCU, C.; IOSIF, D.; MAXIM, D.; BARBĂLATĂ, C.; PALADA, B.; BARBU, I.; BRUDA, I.; IORGULESCU, C.; COCULESCU, L.; SIMIONESCU, P.; PETROFF, M., *Rețea de transmitere și centralizare informații privind transportul și distribuția gazelor naturale*, MP-CCLA, București, 1985.

L-5. LABETOULLE, J.; PUJOLLE, G., *HDLC Throughput and Response Time for Bi-directional Data Flow with Nonuniform Frame Sizes*, IEEE Transactions on Computers, vol. C-30, nr. 6, jun., 1981.

M-1. MACCHI, C.; GUILBERT, J., *Téléinformatique — transport et traitement de l'information dans les réseaux et systèmes téléinformatiques*, DUNOD informatique, Paris, 1983.

M-2. MATYAS, S. M.; MAYER, C. H., *Generation, Distribution and Instalation of Cryptographic Keys*, IBM Syst. J., febr., 1978.

M-3. MERKLE, R. C.; HELLMAN, M. E., *Hiding Information and Signatures in Trapdoor Knapsacks*, IEEE Trans. on Inform. Theory, sept., 1978.

M-4. MOLLOY, M .K., *Performance Analysis Using Stochastic Petri Nets*, IEEE Transactions on Computers, vol. C-31, nr. 9, sept., 1982.

N-1. NEEDHAM, R. M.; SCHROEDER, M., *Using Encryption for Authentication in Large Networks of Computers*, Comm. of ACM., dec., 1978.

P-1. PILAT, F. V.; COCULESCU, L.; CRISTEA, V., *Teleinformatica*, Editura Științifică și Enciclopedică, București, 1986.

P-2. PILAT, F. V., *Clase de modele pentru sisteme de comunicație conținute în Modelul de Referință ISO-OSI*, Comunicare la INFOTEC-86, 22—25 sept., București, 1986.

P-3. PUJOLLE, G.; DROMARD, D.; SERET, D.; HORLAIT, E., *Réseaux et télématique*, Eyrolles, Paris, 1985.

P-4. PRIM, R. C., *Shortest connection networks and some generalisations*, Bell System Technical Journal, vol. 36, 1957.

R-1. RĂDĂCEANU, E., *Limbaje de simulare*, Ed. Militară, București, 1981.

R-2. RENES, A., *Simulation Techniques in Networks Design. Computer Networks and Simulations*, Ed. SCHOEMAKER, S., p. 85—100, N. Holland, 1978.

R-3. RIVEST, R. L.; SHAMIR, A.; ADLEMAN, L., *On Digital Signatures and Public Key Functions*, Comm. of ACM, mai, 1979.

R-4. RUSHDI, A. M., *On Reliability Evaluation by Network Decomposition*, IEEE Transactions on Reability, vol. R-33, dec., 1984.

S-1. SMITH, R. C., *The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver*, IEEE Trans. on Computers, 1980, *C-29*, 12.

S-2. SCHWARTZ, M., *Computer Communication Network. Design and Analysis*, Prentince Hall, Englewood Cliffs, N. J., 1977.

S-3. SCHROEPPEL, R.; SHAMIR, A., *A* $TS = 0(2^n)$ *Time/Space Trade off for Certain NP-Complete Problems*, Proc. of IEEE 20-th Annual Symp . of Found. of Comp. Sci., 1979.

S-4. SHAMIR, A., *On the Crytocomplexity of Knapsack Systems.*, Proc. 11-th Annual ACM Symp. on the Theory of Computing, 1979.

S-5. SIMMONS, G. J., *Symetric and Asymetric Encryption*, ACM., Computing Surveys, dec., 1979.

S-6. SAMARI, N. K.; SCHNEIDER, G. M., *A Queueing Theory-Based Analytic Model of a Distributed Computer Network*, IEEE Transactions on Computers, vol. C-29, nr. 11, nov., 1980.

S-7. SATYANARAYANA, A., *A Unified Formula for Analysis of Some Network Reliability Problems*, IEEE Transactions on Reliability, vol. R-31, apr., 1982.
S-8. SPRAGINS, J., *Dependent Failures in Data Communications Systems*, IEEE Transaction on Communications, vol. COM-25, dec., 1977.
S-9. STEIGLITZ, K.; WEINER, P.; KLEITMAN, D., *The Design of Minimum Cost Survivable Networks*, IEEE Transactions on CT, vol. CT-16, nov., 1969.

Ş-1. ŞERBANAŢI, L. D.; CRISTEA, V., *Metodă de verificare a cooperării între procese în sisteme cu control distribuit, INFO-IAŞI '85.*

T-1. TAKAHASHI, T., *Current INS Model System Operation*, Japan Telecommunications Review, 1986, 28, 2.
T-2. TRANDAFIR, I.; HRIN, R.; MELENCIUC, FL.; RABEGA, I; GHIŞE, E., *Indicaţii metodologice pentru realizarea sistemelor informatice şi a produselor program*, ITCI, Bucureşti, 1987.

V-1. VOSTRÝ, Z.; KRÁLIK, J.; STIEGLER, P.; ZÁVORKA, J. *Program package for dynamic simulation and state reconstruction of gas-distribution pipeline networks*, User's manual, ITIA, ČSAV, Praga, 1985.

ABREVIERI

BDD	— Bază de Date Distribuită
DES	— Data Encryption Standard — Standard de cifrare a datelor
FV	— Fişier virtual
LFSR	— Registru de deplasare cu reacţie liniară
PAR	— Protocol de acces în reţea
PARV	— Protocol de acces la resurse virtuale
SABD	— Sistem de administrare a bazei de date
SARC	— Sistem de administrare a reţelei de calculatoare
SGBD	— Sistem de Gestiune a Bazei de Date
SGBDD	— Sistem de Gestiune a Bazelor de Date Distribuite
SGBDR	— Sistem de gestiune a bazelor de date relaţionale
STI	— Sistem teleinformatic
TV	— Terminal virtual
V.a.c.	— Variabilă aleatoare continuă
V.a.d.	— Variabilă aleatoare discretă

CUPRINS

Prefață 5

Capitolul 1. ARHITECTURA SISTEMELOR TELEINFORMATICE 9
1.1. Concepte de bază 9
1.2. Niveluri arhitecturale 11
1.3. Influența dezvoltării telecomunicațiilor asupra arhitecturii sistemelor teleinformatice 20

Capitolul 2. UTILIZAREA MODELELOR DE CALCUL DISTRIBUIT ÎN PROIECTAREA SISTEMELOR TELEINFORMATICE 25
2.1. Modele algoritmice de calcul paralel 26
2.1.1. Secțiuni critice 27
2.1.2. Regiuni critice 33
2.1.3. Monitoare 37
2.1.4. Rendez-vous 39
2.2. Modele bazate pe transmitere de mesaje 41
2.2.1. Ordonarea evenimentelor în sistemele distribuite 41
2.2.2. Protocol de excludere mutuală 43
2.2.3. Protocol pentru un sistem distribuit de rezolvare ae probleme 46
2.2.4. Protocol bazat pe transmiterea unui privilegiu 49
2.3. Caracteristici ale sistemului de operare relative la gestiunea proceselor 51
2.3.1. Reprezentarea procesorului 51
2.3 2. Reprezentarea procesului 52
2.3.3. Descrierea operațiilor primitive ale proceselor 54
2.3.4. Planificatorul 57
2.3.5. Utilizarea primitivelor de sincronizare pentru comunicarea între procese 58
2.4. Proiectarea protocoalelor pentru sisteme teleinformatice 61

Capitolul 3. UTILIZAREA MODELELOR CU ȘIRURI DE AȘTEPTARE ÎN PROIECTAREA SISTEMELOR TELEINFORMATICE 64
3.1. Modele cu șiruri de așteptare 64
3.1.1. Model. Modelare. Simulare 64
3.1.2. Lege exponențială. Distribuții frecvente în teoria așteptării 65
3.1.3. Notația Kendall a șirurilor de așteptare 68
3.1.4. Șir de așteptare cu o stație de servire (M/M/1) : (FIFO/∞/∞) 69
3.1.5. Rețea de șiruri de așteptare (rețeaua lui Jackson) 70
3.1.6. Rețele BCMP 71
3.1.7. Metode de aproximare 73
3.2. Simulare 74
3.2.1. Clasificare. Etape. Obiective 74
3.2.2. Instrumente de simulare 76
3.2.3. Simularea sistemelor discontinue. Limbajul GPSS 76

3.3. Măsurarea și analiza rezultatelor 78
3.3.1. Statistici furnizate de limbajele de simulare 78
3.3.2. Interpretarea rezultatelor simulării 79
3.3.3. Monitoare de măsură 80
3.4. Aplicații 83
3.4.1. Clase de modele pentru sisteme de comunicație conținute în Modelul de Referință OSI 83
3.4.2. Modelarea și evaluarea factorului de anticipare într-un protocol HDLC 85
3.4.3. Studiul performanțelor unei rețele de calculatoare cu circuite virtuale 87
3.4.4. Model analitic de evaluare a performanțelor unui protocol de comunicații de tip S-W între două noduri ale unei RC cu comutarea de pachete 90
3.5. Considerații privind modelarea comportării STI cu șiruri de așteptare 93

Capitolul 4. TEHNOLOGIA REALIZĂRII UNUI SISTEM TELEINFORMATIC 95

4.1. Noțiuni introductive 95
4.2. Analiză preliminară și specificare funcțională 97
4.2.1. Componentele activității de analiză 97
4.2.2. Tema de realizare 99
4.2.3. Studiu de caz 100
4.3. Proiectare de ansamblu 102
4.3.1. Tehnologia etapei 102
4.3.2. Aspecte privind proiectarea structurilor pentru rețele 108
4.3.3. Probleme în proiectarea rețelelor integrate 117
4.4. Proiectare de detaliu 120
4.4.1. Tehnologia etapei 120
4.4.2. Studiu de caz 122
4.5. Elaborare de programe, integrare și testare 124
4.5.1. Arhitectura dezvoltării de programe pentru un STI 125
4.5.2. Elaborare de programe 128
4.5.3. Pachetul de programe SIMONE 131
4.5.4. Integrare și testare 137
4.5.5. Pachetul de programe NSL 140
4.6. Punere în funcțiune/experimentare și acceptare sistem teleinformatic 142

Capitolul 5. PROIECTAREA SISTEMELOR DE PROTECȚIE ȘI AUTENTIFICARE ÎN STI 145
5.1. Obiective ale protecției informațiilor în STI 145
5.2. Metode criptografice și aplicațiile lor în STI 146
5.2.1. Sisteme criptografice 146
5.2.2. Criptosisteme cu chei publice 148
5.2.3. Semnătura digitală 149
5.3. Proiectarea algoritmului de cifrare computațională 150
5.3.1. Algoritmi de bază 150
5.3.2. Algoritmi produs 151
5.3.3. Algoritmi exponențiali 153
5.3.4. Algoritmi de tip rucsac 156
5.4. Proiectarea sistemelor și protocoalelor criptografice 161
5.4.1. Sisteme criptografice bloc și secvențiale 161
5.4.2. Protecția datelor prin sisteme secvențiale sincrone 161
5.4.3. Protecția datelor prin sisteme secvențiale cu autosincronizare 164
5.4.4. Protecția datelor prin sisteme-bloc 164
5.4.5. Protocoale de protecție și autentificare bazate pe cifruri greu inversabile 167
5.4.6. Integrarea sistemelor criptografice în arhitectura ISO 172

5.5. Proiectarea protocoalelor de administrare și distribuție a cheilor de cifrare în STI 175
5.5.1. Administrarea cheilor în sistemele convenționale 175
5.5.2. Administrarea cheilor publice 179
5.5.3. Generarea cheilor de cifrare bloc 180
5.5.4. Distribuția cheilor de sesiune în rețele deschise 181

Capitolul 6. SISTEME DE PROGRAME PENTRU REALIZAREA APLICAȚIILOR 186

6.1. Comanda terminalelor. Protocol de terminal virtual 186
6.1.1. Modelul terminalului virtual 186
6.1.2. Structura de date a terminalului virtual 188
6.1.3. Analiza unor aplicații 190
6.2. Transferul și prelucrarea fișierelor în STI 193
6.2.1. Atributele de caracterizare ale unui fișier virtual 194
6.2.2. Atributele de prelucrare ale unui fișier virtual 194
6.2.3. Serviciul de fișier virtual 195
6.2.4. Protocoale care utilizează funcțiile serviciului FV 197
6.3. Gestiunea bazelor de date distribuite 200
6.3.1. Noțiuni introductive 200
6.3.2. Caracteristici ale SGBDD din rețele publice 203
6.3.3. Virtualizarea bazelor de date eterogene 205
6.3.4. Arhitectura unei BDD 206
6.3.5. Sincronizarea tranzacțiilor de actualizare 209
6.3.6. Considerații privind proiectarea BDD 210
6.3.7. Exemple de sisteme de gestiune a BDD 213

Capitolul 7. EVALUAREA ȘI OPTIMIZAREA PERFORMANȚELOR STI 216

7.1. Performanțele STI 216
7.2. Analiza disponibilității STI 218
7.2.1. Indicatori de fiabilitate 218
7.2.2. Evaluarea disponibilității STI 225
7.3. Evaluarea timpului de răspuns și a încărcării resurselor 230
7.3.1. Metodologia estimării performanțelor 231
7.3.2. Influența protocoalelor de comunicație asupra performanțelor STI 235
7.3.3. Analiza performanțelor rețelelor distribuite 239
7.4. Optimizarea STI 242
7.4.1. Optimizarea topologiei 243
7.4.2. Optimizarea capacității și a fluxului pe linii 248
7.4.3. Costul STI. Optimizarea STI în funcție de cost 255
7.5. Concluzii 262

Bibliografie selectivă 263
Abrevieri 267

TELEINFORMATIC SYSTEMS DESIGN

CONTENTS

Preface 5

Chapter 1. TELEINFORMATIC SYSTEMS ARCHITECTURE 9

1.1. Basic concepts 9
1.2. Architectural levels 11
1.3. Influence of telecommunications development upon teleinformatic systems (TIS) architecture 20

Chapter 2. USE OF DISTRIBUTED COMPUTING MODELS IN TELEINFORMATIC SYSTEMS DESIGN 25

2.1. Parallel computing algorithmic models 26
2.1.1. Critical sections 27
2.1.2. Critical regions 33
2.1.3. Monitors 37
2.1.4. Rendez-vous 39
2.2. Models based on message transmission 41
2.2.1. Events ordering in distributed systems 41
2.2.2. A mutual exclusion protocol 43
2.2.3. Protocol for distributed system problem solving 46
2.2.4. Protocol based on the transmission of a privilege 49
2.3. Characteristic of operating systems related to process management 51
2.3.1. Processor representation 51
2.3.2. Process representation 52
2.3.3. Describing of the primitive operations of the processes 54
2.3.4. Scheduler 57
2.3.5. Use of synchronization primitives for communication between processes 58
2.4. Teleinformatic system protocol design 61

Chapter 3. USE OF QUEUEING MODELS IN TIS DESIGN 64

3.1. Queueing models 64
3.1.1. Model. Modelling. Simulation 64
3.1.2. Exponential law. Frequent distributions in queueing theory 65
3.1.3. Kendall notation of queueing 68
3.1.4. Queue with a service station (M/M/1) : (FIFO/∞/∞) 69
3.1.5. A queue network (Jackson's network) 70
3.1.6. BCMP networks 71
3.1.7. Approximation methods 73
3.2. Simulation 74
3.2.1. Clasification. Stages. Objectives 74
3.1.2. Simulation tools 76
3.2.3. Discrete systems simulation. CSMP language 76

3.3. Measuring and analysis of the results 78
3.3.1. Statistics provided by simulation languages 78
3.3.2. Interpreting simulation results 79
3.3.3. Measuring monitors 80
3.4. Applications 83
3.4.1. Classes of models for communication systems in ISO-OSI Reference Model 83
3.4.2. Modelling of the anticipation factor in a HDLC protocol 85
3.4.3. Survey of some performances of a virtual circuits computer network 87
3.4.4. Anlythical model of performance estimation of a S-W type communication protocol between two nodes of a packet switching computer network 90
3.5. Conclusions on the modelling of the behaviour of TIS with queues 93

Chapter 4. THE TECHNOLOGY OF DEVELOPING A TIS 95
4.1. Introductory concepts 95
4.2. Preliminary analysis and functional specification 97
4.2.1. Components of analysed activity 97
4.2.2. Development theme 99
4.2.3. Case study 100
4.3. Overall design 102
4.3.1. Stage technology 102
4.3.2. Some aspects concerning the design of the structure for networks 108
4.3.3. Some problems of integrated network design 117
4.4. Detailed design 120
4.4.1. Stage technology 120
4.4.2. Case study 122
4.5. Program elaboration, integration and testing 124
4.5.1. Architecture of program development for a TIS 125
4.5.2. Program elaboration 128
4.5.3. The SIMONE software product 131
4.5.4. Integration and testing 137
4.5.5. The NSL software product 140
4.6. Comissioning/experimentation and acceptance of TIS 142

Chapter 5. DESIGN OF PROTECTION AND AUTHENTIFICATION SYSTEMS IN TIS 145
5.1. Objectives of information protection in TIS 145
5.2. Cryptographic methods and their applications in TIS 146
5.2.1. Cryptographic systems 146
5.2.2. Cryptosystems with public keys 148
5.2.3. Digital signature 149
5.3. Design of encryption algorithm for computational ciphering 150
5.3.1. Basic algorthims 150
5.3.2. Product algorithms 151
5.3.3. Exponential algorithms 153
5.3.4. Knapsack algorithms 156
5.4. Design of cryptographic systems and protocols 161
5.4.1. Block-and sequential-type cryptograpfic systems 161
5.3.2. Data protection by sychronous sequential systems 161
5.4.3. Data protection by selfsynchronous sequential systems 164
5.4.4. Data protection by block systems 164
5.4.5. Protection and authentification protocols based on one-way ciphers .. 167
5.4.6. Cryptographic systems integration in ISO architecture 172

5.5. Design of administration and distribution protocols of ciphering keys in a TIS environment 175
5.5.1. Keys administation in conventional systems 175
5.5.2. Public keys administration 179
5.5.3. Block ciphering keys generation 180
5.5.4. Session keys distribution in open network 181

Chapter 6. SOFTWARE SYSTEMS USED IN CARRYING OUT OF APPLICATIONS 186

6.1. Terminals command. Virtual-terminal protocol (VTP) 186
6.1.1. Virtual terminal model 186
6.1.2. Virtual terminal data structure 188
6.1.3. Analysis of some applications 190
6.2. Transfer and file processing in TIS 193
6.2.1. Characterization atributes of a virtual file (VF) 194
6.2.2. Processing atributes of a virtual file 194
6.2.3. Virtual file service 195
6.2.4. Protocols using the functions of VF services 197
6.3. Distributed Data Base (DDB) Management 200
6.3.1. Introductory concepts 200
6.3.2. Characteristics of a Distributed Data Base Management System (DDBMS) 203
6.3.3. Virtualization of heterogenous data bases 205
6.3.4. Architectures of a DDB 206
6.3.5. Synchronization of up-dating transactions 209
6.3.6. Considerations concerning the design of a DDB 210
6.3.7. Exemples of DDBMS 213

Chapter 7. PERFORMANCES ESTIMATION AND OPTIMIZATION OF TIS 216

7.1. Performances of a TIS 216
7.2. Disponibility analysis of a TIS 218
7.2.1. Reliability indicators 218
7.2.2. TIS disponibility evaluation 225
7.3. Time-response and resources-loading evaluation 230
7.3.1. Performance estimation methodology 231
7.3.2. Communication protocols influence upon TIS performances 235
7.3.3. Performance analysis of distributed networks 239
7.4. TIS optimization 242
7.4.1. Topology optimization 243
7.4.2. Capacity and line-flow optimization 248
7.4.3. TIS cost. TIS optimization vs. cost 255
7.5. Conclusions 262

Selected papers 263

List of abbreviations 267

CONCEPTIONS DES SYSTEMES TELEINFORMATIQUES

TABLE DE MATIERES

Avant-propos 5

1. L'ARCHITECTURE DES SYSTEMES TELEINFORMATIQUES 9
   1.1. Concepts de base 9
   1.2. Niveaux architecturaux 11
   1.3. L'influence du développement des télécommunications sur l'architecture des systèmes téléinformatiques 20

2. L'UTILISATION DES MODELES DE CALCUL DISTRIBUE DANS LA CONCEPTION DES SYSTEMES TÉLEINFORMATIQUES 25
   2.1. Modèles algorithmiques de calcul parallèle 26
      2.1.1. Sections critiques 27
      2.1.2. Régions critiques 33
      2.1.3. Moniteurs 37
   2.2. Des modèles basés sur la transmission de messages 41
      2.2.1. Déroulement des événements dans les systèmes distribués 41
      2.2.2. Un protocole d'exclusion mutuelle 43
      2.2.3. Protocole pour un système distribué destiné à résoudre des problèmes 46
      2.2.4. Protocole basé sur la transmission d'un privilège 49
   2.3. Caractéristiques du système d'exploitation relatives à la gestion des processus 51
      2.3.1. Représentation du processeur 51
      2.3.2. Représentation du processus 52
      2.3.3. Description des opérations primitives des processus 54
      2.3.4. Le planificateur 57
      2.3.5. Utilisation des primitives de synchronisation pour la communication entre les processus 58
   2.4. Conception des protocoles pour les systèmes téléinformatiques 61

3. L'UTILISATION DES MODELES AVEC DES FILES D'ATTENTE DANS LA CONCEPTION DES SYSTEMES TELEINFORMATIQUES 64
   3.1. Modèles avec des files d'attente 64
      3.1.1. Modèle. Modélage. Simulation 64
      3.1.2. Loi exponentielle. Distribution des fréquences dans la théorie d'attente 65
      3.1.3. Notation Kendall des files d'attente 68
      3.1.4. File d'attente avec une station de service (M/M/1) : (FIFO) ∞/∞ .. 69
      3.1.5. Réseau de files d'attente (réseau de Jackson) 70
      3.1.6. Réseaux BCMP 71
      3.1.7. Méthodes d'approximation 73
   3.2. Simulation 74
      3.2.1. Classification. Etapes. Objectifs 74

3.2.2. Instruments de simulation 76
3.2.3. Simulation des systèmes discretes. Le langage GPSS 76
3.3. Mesurage et analyse des résultats 78
3.3.1. Statistiques fournies par les langages de simulation 78
3.3.2. Interprétation des résultats de la simulation 79
3.3.3. Moniteurs de mesure 80
3.4. Applications 83
3.4.1. Classes de modèles pour le système de communications inclus dans le modèle de référence ISO-OSI 83
3.4.2. Modélage du facteur d'anticipation dans le protocole HDLC 85
3.4.3. Etude de quelques performances d'un réseau d'ordinateurs avec des circuits virtuels 87
3.4.4. Modèle analytique d'évaluation des performances d'un protocole de communications de type S-W entre deux noeuds d'une RC avec commutation de paquets 90
3.5. Conclusions visant le modélage du comportement du système téléinformatique STI avec des files d'attente 93

4. TECHNOLOGIE DE LA REALISATION D'UN SYSTEME TELEINFORMATIQUE 95

4.1. Introduction 95
4.2. Analyse préliminaire et spécification fonctionnelle 97
4.2.1. Structure de l'activité d'analyse 97
4.2.2. Thème de réalisation 99
4.2.3. Etude de cas 100
4.3. Conception d'ensemble 102
4.3.1. Technologie de l'étape 102
4.3.2. Aspects visant la conception des structures pour les réseaux 108
4.3.3. Problèmes dans la conception des réseaux intégrés 117
4.4. Conception de détail 120
4.4.1. Technologie de l'étape 120
4.4.2. Etude de cas 122
4.5. Elaboration de programmes, l'étape d'integration et de test 124
4.5.1. Architecture du développement de programmes pour le STI 125
4.5.2. Elaboration de programmes 128
4.5.3. Le paquet de programmes SIMONE 131
4.5.4. Integration et test 137
4.5.5. Le paquet de programmes NSL 140
4.6. Mise en fonction/expérimentation et acceptation de système téléinformatique 142

5. LA CONCEPTION DES SYSTEMES DE PROTECTION ET D'AUTHENTIFICATION EN STI 145

5.1. Objectifs de la protection des informations en STI 145
5.2. Méthodes cryptographiques et leurs applications en STI 146
5.2.1. Cryptosystèmes 146
5.2.2. Cryptosystèmes à clés publiques 148
5.2.3. Signature digitale 149
5.3. Conception de l'algorithme de cryptage computationnel 150
5.3.1. Algorithmes de base 150
5.3.2. Algorithmes produits 151
5.3.3. Algorithmes exponentiels 153
5.3.4. Algorithmes à empilement 156
5.4. Conception des systèmes et des protocoles cryptographiques 161
5.4.1. Cryptosystèmes blocs et séquentiels 161
5.4.2. Protection des données par des systèmes séquentiels synchrones 161
5.4.3. Protection des données par des systèmes séquentiels à autosynchronisation 164
5.4.4. Protection des données par des systèmes blocs 164
5.4.5. Protocoles de protection et d'authentification basés sur des chiffres noninversibles 167

5.4.6. Cryptographie et architecture ISO 172
5.5. Conception des protocoles de gestion et de distribution des clés de cryptage en STI 175
5.5.1. Gestion des clés dans les systèmes conventionnels 175
5.5.2. Gestion des clés publiques 179
5.5.3. Génération des clés de cryptage bloc 180
5.5.4. Distribution des clés dans les réseaux ouverts 181

6. SYSTEMES DE PROGRAMMES POUR LA REALISATION DES APPLICATIONS 186
6.1. Commande des terminaux. Protocole de terminal virtuel 186
6.1.1. Modèle des terminal virtuel 186
6.1.2. Structure des données du terminal virtuel 188
6.1.3. Analyse de quelques applications 190
6.2. Transfert et traitement des fichiers en STI 193
6.2.1. Les attributs qui caractérisent un fichier virtuel 194
6.2.2. Les attributs des traitements d'un fichier virtuel 194
6.2.3. Le service de fichier virtuel 195
6.2.4. Des protocoles qui utilisent les fonctions du service FV 197
6.3. Gestion des bases de données distribuées (BDD) 200
6.3.1. Introduction 200
6.3.2. Caractéristiques des BDD des réseaux publiques 203
6.3.3. Virtualisation des bases de données étérogènes 205
6.3.4. Architecture d'une BDD 206
6.3.5. Synchronisation des transactions d'actualisation 209
6.3.6. Considérations visant la conception de la BDD 210
6.3.7. Exemples de systèmes de gestion de la BDD 213

7. EVALUATION ET OPTIMISATION DES PERFORMANCES DES STI 216
7.1. Performances des STI 216
7.2. Analyse de la disponibilité des STI 218
7.2.1. Indicateurs de fiabilité 218
7.2.2. Evaluation de la disponibilité des STI 225
7.3. Evaluation du temps de réponse et du chargement des ressources 230
7.3.1. Méthodologie de l'estimation des performances 231
7.3.2. Influence des protocoles de communication sur les performances des STI 235
7.3.3. Analyse des performances des réseaux distribués 239
7.4. Optimisation des STI 242
7.4.1. Optimisation de la tipologic 243
7.4.2. Optimisation de la capacité et du flux sur les lignes 248
7.4.3. Coût des STI. Optimisation des STI en fonction du coût 255
7.5. Conclusions 262

Bibliographie sélective 263

Abréviations 267

ПРОЕКТИРОВАНИЕ СИСТЕМ ТЕЛЕПЕРЕДАЧИ ИНФОРМАЦИИ (СТПИ)

СОДЕРЖАНИЕ

Предисловие 5

Глава 1. АРХИТЕКТУРА СИСТЕМ ТЕЛЕПЕРЕДАЧИ ИНФОРМАЦИИ (СТПИ) 9

1.1. Основные концепты 9
1.2. Уровни архитектуры 11
1.3. Влияние развития телесвязей на архитектуру СТПИ 20

Глава 2. ИСПОЛЬЗОВАНИЕ МОДЕЛЕЙ РАССПРЕДЕЛЕННОГО ВЫЧИСЛЕНИЯ ПРИ ПРОЕКТИРОВАНИИ СТПИ 25

2.1. Алгоритмические модели параллельного вычисления 26
2.1.1.Критические рассечения 27
2.1.2. Критические области 33
2.1.3. Мониторы 37
2.1.4. Встреча 39
2.2. Модели основаны на передаче сообщений 41
2.2.1. Упорядочение событий в расспределенных системах..... 41
2.2.2. Протокол совместного исключения 43
2.2.3. Протокол для расспределенной системы решения задач 46
2.2.4. Протокол основан на передаче привилегии 49
2.3. Характеристики операционной системы связанные с управлением процессами 51
2.3.1. Изображение процессора 51
2.3.2. Изображение процесса 52
2.3.3. Описание примитивных операций процессов 54
2.3.4. Планификатор 57
2.3.5. Использование примитивов синхронизации для сообщения между процесасми 58
2.4. Проектирование протоколов для СТПИ 61

Глава 3. ИСПОЛЬЗОВАНИЕ МОДЕЛЕЙ С ЦЕПЯМИ ОЖИДАНИЯ В ПРОЕКТИРОВАНИИ СТПИ 64

3.1. Модели с цепями ожидания 64
3.1.1. Модель. Моделирование. Имитация 64
3.1.2. Экспоненцияльный закон. Частые расспределения в теории ожидания 65
3.1.3. Нотация Кендаля цепей ожидания 68
3.1.4. Цепь ожидания со сервисной станцией (M/M/I): (FIFO /∞/∞).... 69
3.1.5. Сеть цепей ожидания (сеть Джексона) 70
3.1.6. Сети BCMP 71
3.1.7. Методы приблизительной оценки 73
3.2. Имитация 74
3.2.1. Классификация. Этапы. Цели 74

3.2.2. Инструменты для имитации 76
3.2.3. Имитация дискретных систем. GPSS 76
3.3. Измерение и анализ результатов 78
3.3.1. Статистики поставляемые языками имитации 78
3.3.2. Интерпретация результатов имитации 79
3.3.3. Мониторы измерения 80
3.4. Применения 83
3.4.1. Классы моделей для системы сообщении содержаной в Ориентировочной Модели 83
3.4.2. Моделирование фактора предвохсищения в протоколе HDLC 85
3.4.3. Изучение рекордных показателей сети ЭВМ с виртуальными цепями 87
3.4.4. Аналитическая модель оценки рекордных показателей протокола сообщений типа S-W между двумя узлами сети ЭВМ с пакетным переключением 90
3.5. Выводы насчет моделирования образа действия СТПИ с цепями ожидания 93

Глава 4. ТЕХНОЛОГИЯ РЕАЛИЗАЦИИ СТПИ 95
4.1. Вводные понятия 95
4.2. Предварительный анализ и функциональная спецификация 97
4.2.1. Компоненты анализа 97
4.2.2. Тема реализации 99
4.2.3. Исследование случая 100
4.3. Общее проектирование 102
4.3.1. Технология этапа 102
4.3.2. Виды проектирования структур для сетей 108
4.3.3. Проблемы в проектировании интегрированных сетей 117
4.4. Детальное проектирование 120
4.4.1. Технология этапа 120
4.4.2. Исследование случая 122
4.5. Разработка программ, интеграция и тестирование 124
4.5.1. Архитектура развития программ для одной СТПИ 125
4.5.2. Разработка программ SIMONE 128
4.5.3. Пакет программ 131
4.5.4. Интеграция и тестирование 137
4.5.5. Пакет программ NSL 140
4.6. Пуск в эксплуатацию/экспериментирование и принятие СТПИ 142

Глава 5. ПРОЕКТИРОВАНИЕ СИСТЕМ ЗАЩИТЫ И ЗАСВИДЕТЕЛЬСТВОВАНИЯ В СТПИ 145
5.1. Цели защиты информации в СТПИ 145
5.2. Криптографические методы и их применения в СТПИ 146
5.2..1 Криптографические системы 146
5.2.2. Криптосистемы с публичными ключами 148
5.2.3. Дигитальная подпись 149
5.3. Проектирование алгоритма компютациональной зашифровки 150
5.3.1. Основные алгоритмы 150
5.3.2. Алгоритмы-продукты 151
5.3.3. Экспоненциальные алгоритмы 153
5.3.4. Алгоритмы типа рюкзак 156
5.4. Проектирование криптографических систем и протоколов 161
5.4.1. Криптографические системы типа блок и последовательные 161
5.4.2. Защита данных через синхронные последовательные системы 161
5.4.3. Защита данных через последовательные системы с самосинхронизацией 164
5.4.4. Защита данных через системы типа блок 164

5.4.5. Протоколы защиты и засвидетельствования основаны на тяжело инверсируемых шифрах 167
5.4.6. Интеграция криптографических систем в архитектуре ISO 172
5.5. Проектирование протоколов управления и расспределения ключей шифровки в СТПИ 175
5.5.1. Управление ключами в условных системах 175
5.5.2. Управление публичными ключами 179
5.5.3. Генерирование ключей шифровки типа блок 180
5.5.4. Расределение сеансовых ключей в открытых сетях 181

Глава 6. СИСТЕМЫ ПРОГРАММ ДЛЯ РЕАЛИЗАЦИИ ПРИМЕНЕНИЙ 186

6.1. Команда терминалов. Протокол виртуального терминала 186
6.1.1. Модель виртуального терминала 186
6.1.2. Структура данных виртуального терминала 188
6.1.3. Анализ применений 190
6.2. Перенос и обработка файлов в СТПИ 193
6.2.1. Атрибуты характеризации виртуального файла 194
6.2.2. Артибуты обработки виртуального файла 194
6.2.3. Услуги виртуального файла 195
6.2.4. Протоколы которые используют функции услуги ВФ 197
6.3. Управление распределенными базами данных 200
6.3.1. Вводные понятия 200
6.3.2. Характеристики СУРБД публичных сетей 203
6.3.3. Виртуализация разнородных баз данных 205
6.3.4. Архитектура распределенной базы данных 206
6.3.5. Синхронизация актуализирующих транзакций 209
6.3.6. Соображения о проектировании распределенных банках данных 210
6.3.7. Примеры СУРБД 213

Глава 7. ОЦЕНКА И ОПТИМИЗАЦИЯ РЕКОРДНЫХ ПОКАЗАТЕЛЕЙ СТПИ 216

7.1. Рекордные показатели СТПИ 216
7.2. Анализ возможностей СТПИ 218
7.2.1. Показатели надежности 218
7.2.2. Оценка возможностей СТПИ 225
7.3. Оценка времени ответа и загрузки ресурсов 230
7.3.1. Методология оценки рекордных показателей 231
7.3.2. Влияние протоколов сообщения на рекордные показатели СТПИ 235
7.3.3. Анализ рекордных показателей распределенных сетях 239
7.4. Оптимизация СТПИ 242
7.4.1. Оптимизация топологии 243
7.4.2. Оптимизация емкости и потока на линиях 248
7.4.3. Стоимость СТПИ. Оптимизация СТПИ зависимо от стоимости 255
7.5. Заключение 262

Библиография 263

Список сокращений 267

Utilizarea rețelelor de calculatoare, care constituie suportul sistemelor teleinformatice, a permis rezolvarea unor probleme greu de abordat cu alte mijloace, pentru care accesul la distanță și distribuirea puterii de calcul reprezintă condiții esențiale: conducerea producției, dirijarea traficului, supravegherea distribuției energiei și a combustibililor, rezervarea locurilor etc. Totodată, perfecționarea rețelelor de calculatoare a stimulat utilizatorii, generînd noi aplicații cum sînt cele telematice, orientate spre probleme curente ale publicului larg. O astfel de perfecționare nu ar fi fost posibilă fără o abordare sistematică și bine fundamentată teoretic a conceperii și realizării sistemelor teleinformatice, putîndu-se astăzi vorbi de o tehnologie specifică acestor sisteme. Unele din aspectele sale cele mai semnificative sînt cuprinse în lucrarea de față.